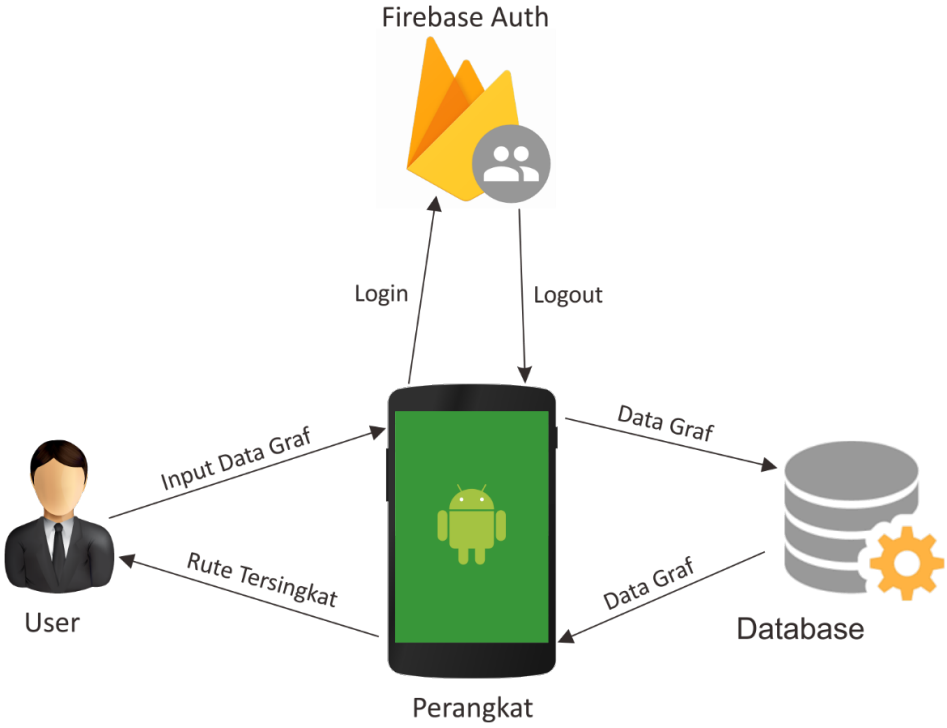
BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan

Pada tahapan perancangan di setiap *increment* yang dilakukan, digunakan pendekatan desain berorientasi objek yang direpresentasikan menggunakan UML (*Unified Modelling Languange*). Terdapat beberapa tahap pengerjaan yang dilakukan, yaitu perancangan arsitektur sistem, perancangan *sequence* diagram, perancangan pemodelan relasional data model, perancangan algoritma dan perancangan antarmuka.

5.1.1 Perancangan Arsitektural *Increment* 1

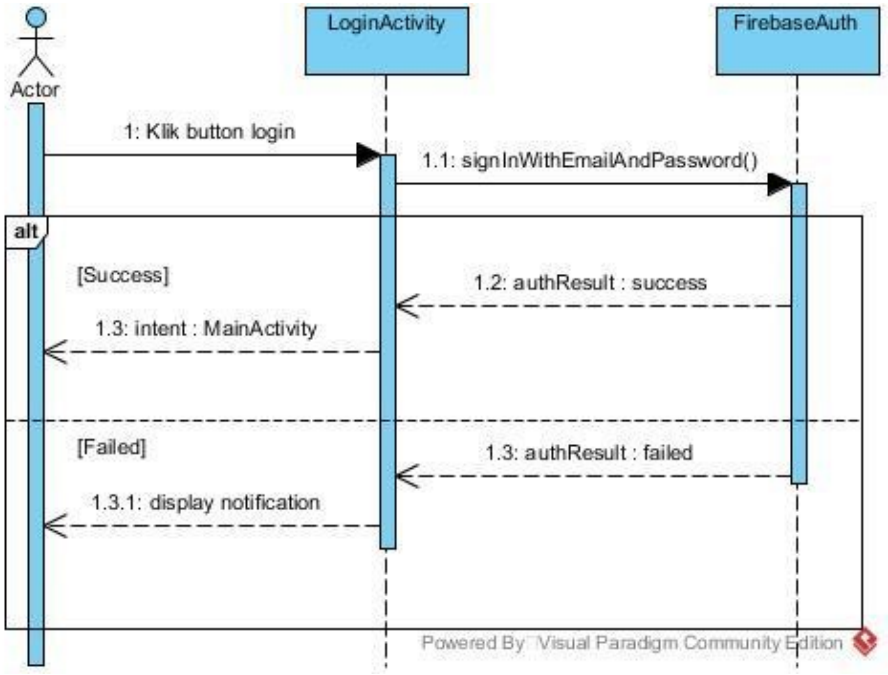
Sistem Aplikasi Mobile Struktur Data Graf berjalan pada perangkat *mobile* dengan sistem operasi Android. Pengguna harus terautentikasi terlebih dahulu untuk menjalankan fungsionalitas aplikasi. Proses autentikasi pengguna dilakukan menggunakan *backend service* Firebase Authentication yang dapat mengautentikasi pengguna menggunakan *email* dan *password* yang telah didaftarkan sebelumnya. Autentikasi yang berhasil nantinya akan membuka akses fungsionalitas aplikasi kepada pengguna. Pengguna berinteraksi dengan perangkat untuk memasukkan data graf yang terdiri atas titik-titik yang saling terhubung membentuk jalur. Data graf ini akan disimpan dalam database lokal Android dengan menggunakan sistem basis data SQLite dan hanya bisa diakses oleh pengguna yang telah terautentikasi. Data graf yang disimpan ini nantinya akan diuji dengan mengimplementasikannya menggunakan algoritma Djikstra untuk mendapatkan rute tersingkat dari suatu lokasi ke salah satu lokasi di Universitas Brawijaya. Perancangan Arsitektural Sistem pada *increment* 1 dapat dilihat pada Gambar 5.1



Gambar 5.1 Perancangan Arsitektural *Increment* 1

5.1.2 Perancangan *Sequence* Diagram *Increment* 1

5.1.2.1 *Sequence* Diagram Login

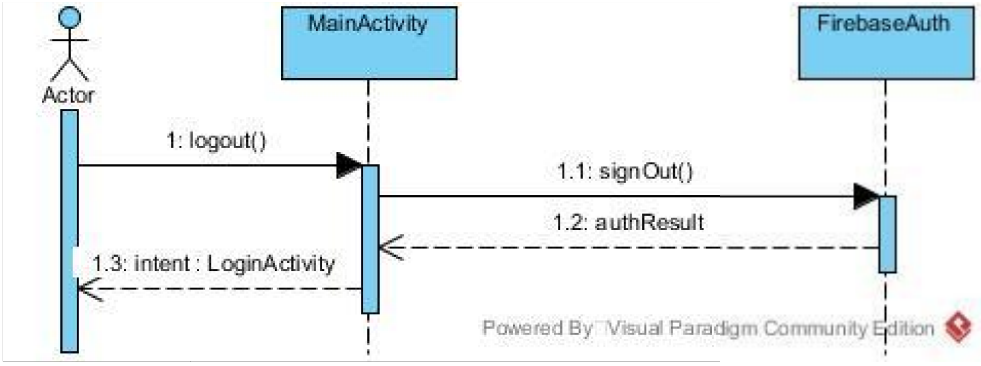


Gambar 5.2 *Sequence* Diagram Login *Increment* 1

Pada Gambar 5.2, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Login. Selain itu, terdapat alur yang terjadi dalam menjalankan fungsi tersebut.

Pada *Sequence* Diagram Login, aktor memasukkan data berupa *email* dan *password*. Setelah data diisi, aktor menekan tombol *login* untuk melakukan proses *login*. Kemudian sistem akan memanggil method signInWithEmailAndPassword() dari kelas FirebaseAuth untuk mengautentikasi user ke dalam aplikasi. Jika kembalian dari FirebaseAuth yaitu authResult bernilai *success*, maka aktor akan mendapatkan *intent* yang akan mengubah tampilan ke halaman MainActivity untuk membuka akses fungsionalitas aplikasi kepada pengguna.Jika kembalian dari FirebaseAuth yaitu authResult bernilai *failed*, maka aktor akan mendapatkan *intent* yang akan memunculkan notifikasi kesalahan.

5.1.2.2 *Sequence* Diagram Logout

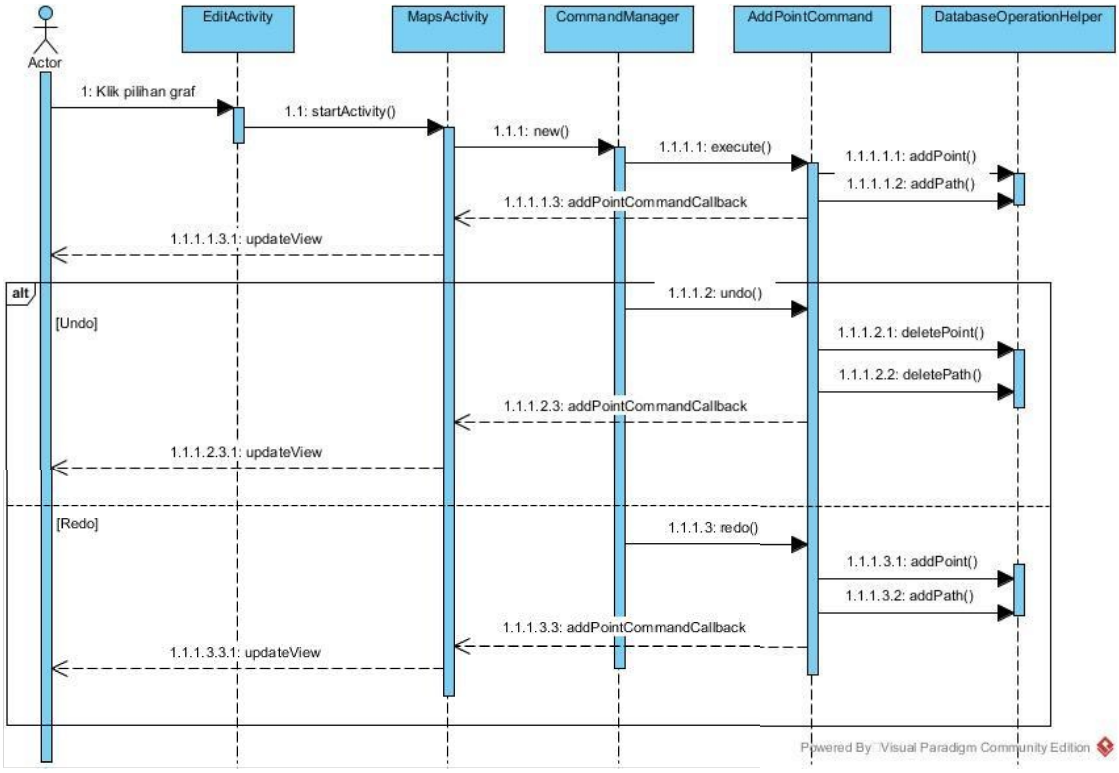


Gambar 5.3 *Sequence* Diagram Logout *Increment* 1

Pada Gambar 5.3, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Logout. Selain itu, terdapat alur yang terjadi dalam menjalankan fungsi tersebut.

Pada *Sequence* Diagram Logout, aktor menekan tombol *logout* untuk keluar dari aplikasi. Setelah itu, sistem akan memanggil method signOut() dari kelas FirebaseAuth untuk mengeluarkan aktor dari sistem. Setelah itu, aktor akan dipindahkan ke halaman LoginActivity.

5.1.2.3 *Sequence* Diagram Membuat Titik Dalam Graf



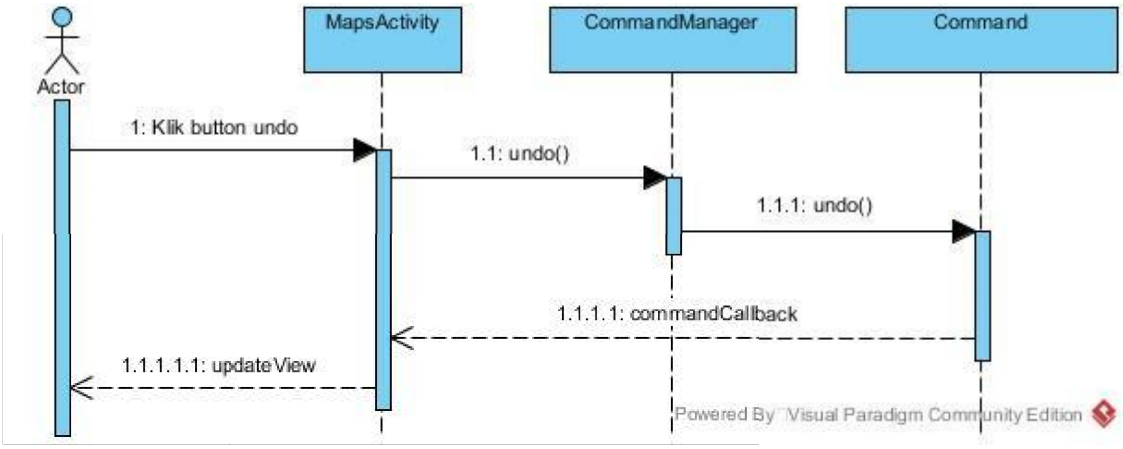
Gambar 5.4 *Sequence* Diagram Membuat Titik Dalam Graf *Increment* 1

Pada Gambar 5.4, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Membuat Titik Dalam Graf. Selain itu, terdapat alur yang terjadi dalam menjalankan fungsi tersebut.

Pada *Sequence* Diagram Membuat Titik Dalam Graf, pengguna memilih terlebih dahulu graf yang akan di edit pada EditActivity. Setelah itu, sistem akan memanggil MapsActivity beserta variable pilihan data yang dipilih oleh aktor. Fungsionalitas mengedit graf menggunakan *Command Pattern* dalam menjalankan fungsi-fungsi nya. Pada konteks ini, kelas MapsActivity bertindak sebagai *client*¸ kelas CommandManager bertindak sebagai *invoker*, kelas DatabaseOperationHelper bertindak sebagai *receiver*, dan kelas AddPointCommand bertindak sebagai *Concrete Command* yang merupakan *child* dari kelas Command. MapsActivity akan membuat instans CommandManager, setelah itu instans CommandManager ini akan digunakan untuk memanggil method execute, undo, dan redo pada setiap Command yang terdapat pada stack di dalam CommandManager. Aksi *execute* akan membuat *point* dan *path* baru pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi database. Aksi *undo* akan menghapus *point* dan *path* yang telah ditambahkan pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Aksi *redo* akan memunculkan kembali *point* dan *path* yang terhapus ketika aksi *undo* dijalankan dengan membuat *point* dan *path* baru dengan menggunakan kelas

DabaseOperationHelper sesuai dengan data yang ada pada aksi *execute*. Setiap aksi yang dilakukan (*execute*, *undo*, dan *redo*) akan memicu *callback* addPointCommandCallback yang akan melakukan pembaruan terhadap perubahan yang terjadi ke tampilan peta.

5.1.2.4 *Sequence* Diagram Undo

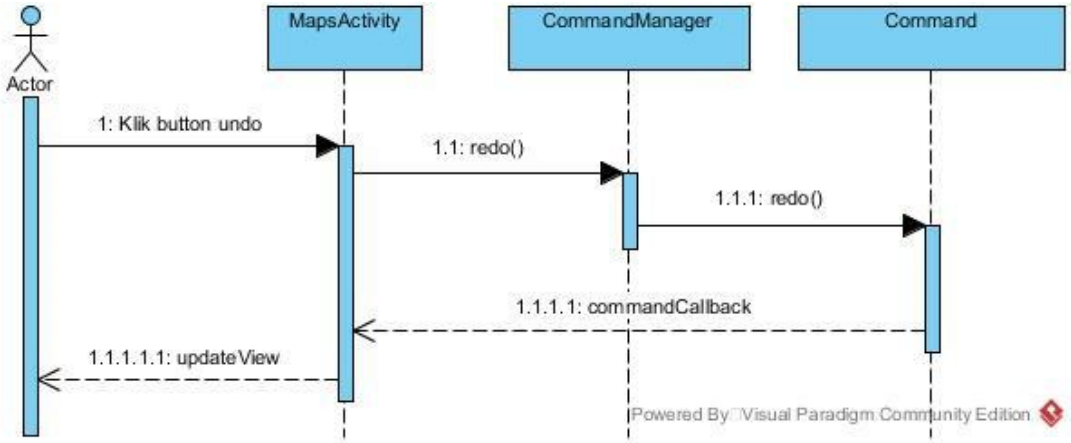


Gambar 5.5 Sequence Diagram Undo

Pada Gambar 5.4, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Undo. Selain itu, terdapat alur yang terjadi dalam menjalankan fungsi tersebut.

Pada *Sequence* Diagram Undo, pengguna melakukan *tap* pada tombol *undo* untuk mengembalikan kondisi graf ke aksi sebelumnya. Tombol *undo* akan memanggil kelas CommandManager sebagai kelas *invoker* untuk melakukan fungsi *undo*. Kemudian, CommandManager akan memicu Command pada *stack* teratas nya untuk menjalankan fungsi *undo*. Selanjutnya, Command yang melakukan fungsi *undo* akan mengembalikan *callback* ke MapsActivity untuk melakukan *update* terhadap *view* pengguna.

5.1.2.5 *Sequence* Diagram Redo

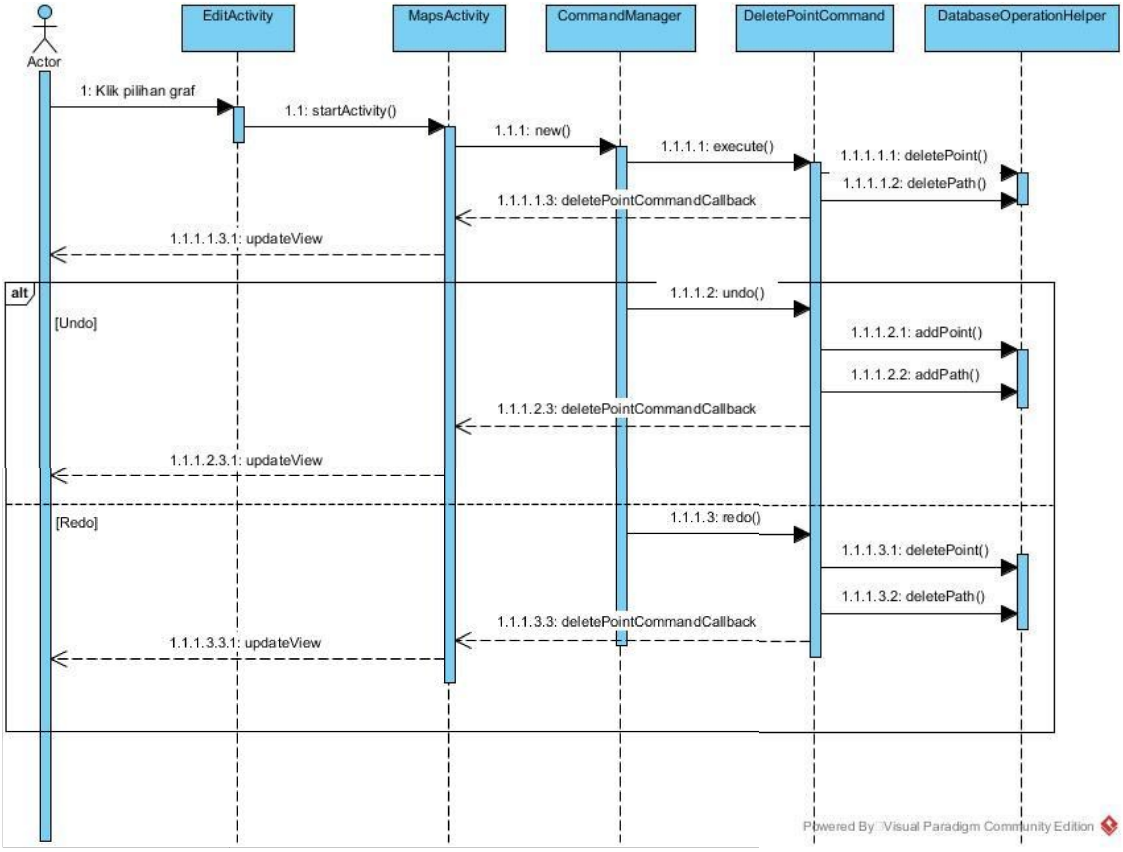


Gambar 5.6 Sequence Diagram Redo

Pada Gambar 5.4, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Redo. Selain itu, terdapat alur yang terjadi dalam menjalankan fungsi tersebut.

Pada *Sequence* Diagram Undo, pengguna melakukan *tap* pada tombol *redo* untuk mengembalikan kondisi graf ke aksi sebelum aksi *undo* dilakukan. Tombol *redo* akan memanggil kelas CommandManager sebagai kelas *invoker* untuk melakukan fungsi *redo*. Kemudian, CommandManager akan memicu Command pada *stack* teratas nya untuk menjalankan fungsi *redo*. Selanjutnya, Command yang melakukan fungsi *redo* akan mengembalikan *callback* ke MapsActivity untuk melakukan *update* terhadap *view* pengguna.

5.1.2.6 *Sequence* Diagram Menghapus Titik



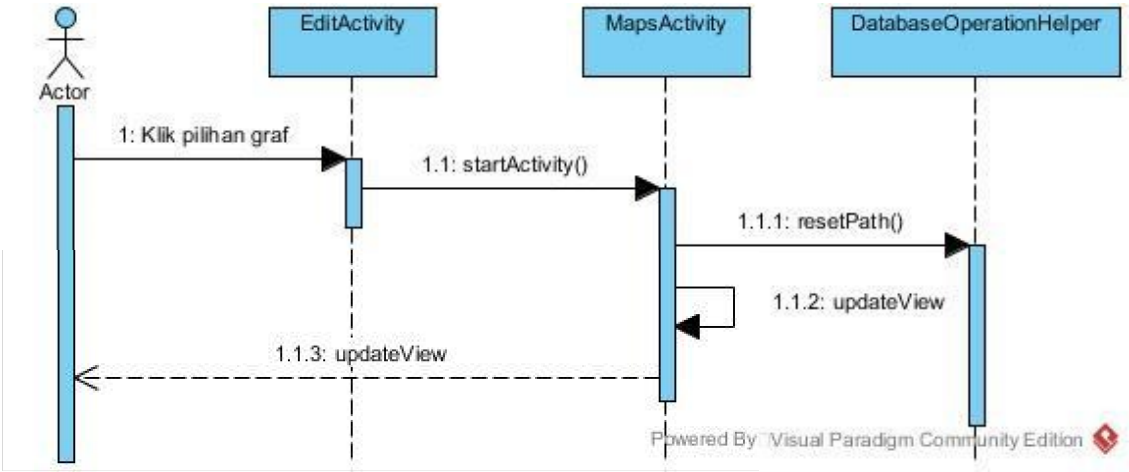
Gambar 5.7 Sequence Diagram Menghapus Titik *Increment* 1

PadaGambar 5.7, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Menghapus Titik. Selain itu, terdapat alur yang terjadi dalam menjalankan fungsi tersebut.

Pada *Sequence* Diagram Menghapus Titik, pengguna memilih terlebih dahulu graf yang akan di edit pada EditActivity. Setelah itu, sistem akan memanggil MapsActivity beserta variable pilihan data yang dipilih oleh aktor. Fungsionalitas mengedit graf menggunakan *Command Pattern* dalam menjalankan fungsi-fungsi nya. Pada konteks ini, kelas MapsActivity bertindak sebagai *client*¸ kelas CommandManager bertindak sebagai *invoker*, kelas DatabaseOperationHelper bertindak sebagai *receiver*, dan kelas DeletePointCommand bertindak sebagai *Concrete Command* yang merupakan *child* dari kelas Command. MapsActivity akan membuat instans CommandManager, setelah itu instans CommandManager ini akan digunakan untuk memanggil method *execute*, *undo*, dan *redo* pada setiap Command yang terdapat pada stack di dalam CommandManager. Aksi *execute* akan menghapus *point* dan *path* yang ditambahkan pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Aksi *undo* akan membuat kembali *point* dan *path* yang telah terhapus ketika aksi *execute* dijalankan dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi database. Aksi *redo* akan menghapus kembali *point* dan *path*

yang telah dikembalikan ketika aksi *undo* dijalankan dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Setiap aksi yang dilakukan (*execute*, *undo*, dan *redo*) akan memicu *callback* addPointCommandCallback yang akan melakukan pembaruan terhadap perubahan yang terjadi ke tampilan peta.

5.1.2.7 *Sequence* Diagram Reset Graf

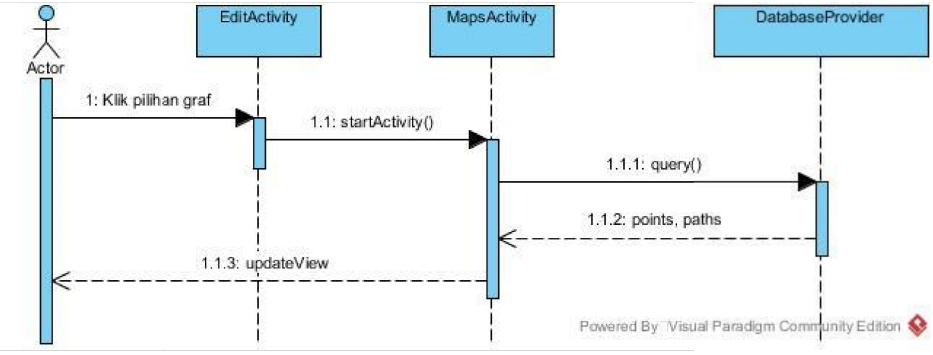


Gambar 5.8 Sequence Diagram Reset Graf *Increment* 1

Pada Gambar 5.8, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Reset Graf. Selain itu, terdapat alur yang terjadi dalam menjalankan fungsi tersebut.

Pada *Sequence* Diagram Reset Graf, aktor memilih terlebih dahulu jenis graf yang akan di reset. Setelah itu sistem akan menampilkan halaman MapsActivity. Pada MapsActivity, terdapat method yang akan memanggil method resetPath() pada kelas DatabaseOperationHelper yang akan menghapus seluruh titik dan jalur pada graf yang dipilih. Setelah itu, MapsActivity akan melakukan updateView untuk menghilangkan jalur dan titik yang ada pada tampilan peta.

5.1.2.8 *Sequence* Diagram Melihat Graf

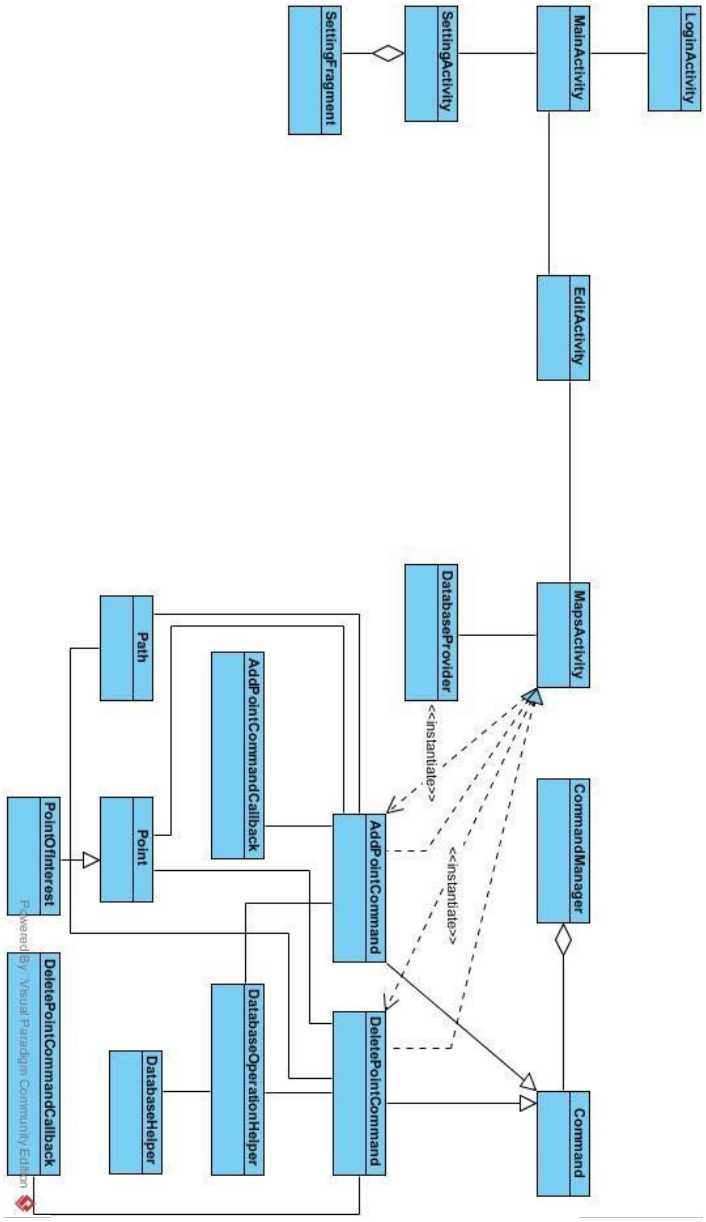


Gambar 5.9 *Sequence* Diagram Melihat Graf *Increment* 1

Pada Gambar 5.9, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Melihat Graf. Selain itu, terdapat alur yang terjadi dalam menjalankan fungsi tersebut.

Pada *Sequence* Diagram Logout, aktor memilih terlebih dahulu jenis graf yang akan di *reset*. Setelah itu sistem akan menampilkan halaman MapsActivity. Pada MapsActivity, terdapat *method* yang akan memanggil method query() untuk melakukan pengambilan data dari *database* menggunakan DatabaseProvider. DatabaseProvider akan memberikan data graf yang diminta, selanjutnya MapsActivity merubah *view* nya dengan menambah *marker* dan *polyline* sesuai data graf yang didapatkan.

5.1.3 Perancangan *Class* Diagram *Increment* 1



Gambar 5.10 Rancangan Kelas Diagram *Increment* 1

Pada rancangan kelas diagram *Increment* 1 yang digambarkan pada Gambar 5.10, terdapat 5 kelas Activity dan 1 kelas Fragment yang berinteraksi dengan pengguna. Penjelasan fungsi kelas-kelas tersebut dapat dilihat pada Tabel 5.1.

Tabel 5.1 Penjelasan Fungsi Kelas Yang Berinteraksi Dengan Pengguna *Increment* 1

Nama Kelas	Fungsi
LoginActivity	Kelas yang berinteraksi dengan pengguna untuk menajalankan fungsi Login.

MainActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan halaman utama dan pilihan menu yang dapat di akses oleh pengguna.
EditActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan pilihan graf yang akan di ubah.
MapsActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan tampilan peta beserta graf yang telah dipilih oleh pengguna dan pilihan untuk menambah titik, menghapus titik, dan reset graf.
SettingActivity	Kelas yang menampung halaman untuk setting.
SettingFragment	Kelas yang berinteraksi dengan pengguna untuk menampilkan halaman *setting*.

Kemudian terdapat kelas yang dibuat dengan mengikuti pola perancangan *Command Pattern* yang berperan sebagai *client, receiver, invoker, command,* dan *concrete command*. Implementasi pola perancangan *Command Pattern* pada perancangan kelas diagram *increment* 1 dapat dilihat pada Tabel 5.2.

Tabel 5.2 Penjelasan Fungsi Kelas Yang Mengikut Pola Perancangan *Command Pattern Increment* 1

Peran	Implementasi
Client	Kelas yang menjadi *client* dalam pola perancangan Command Pattern adalah kelas MapsActivity. Berfungsi untuk melakukan permintaan perubahan menggunakan *Concrete Command*.
Receiver	Kelas yang menjadi *receiver* dalam pola perancangan *Command Pattern* adalah kelas DatabaseOperationHelper. Berfungsi untuk melakukan perubahan *database*.
Invoker	Kelas yang menjadi *invoker* dalam pola perancangan *Command Pattern* adalah kelas *CommandManager*. Berfungsi untuk menjalankan perintah yang diminta oleh *client*.
Command	Kelas yang menjadi *command* dalam pola perancangan *Command Pattern* adalah kelas *Command*. Berfungsi sebagai *interface* untuk *Concrete Command*.
Concrete Command	Kelas yang menjadi *concrete command* pada pola perancangan *Command Pattern* adalah kelas AddPointCommand dan DeletePointCommand.

	Kelas ini berfungsi untuk mendefinisikan perintah yang tersedia. Setiap *concrete command* memiliki *callback* masing-masing yang digunakan untuk melakukan perubahan tampilan ketika operasi *database* berhasil dilakukan.

Kelas lain yang terdapat pada perancangan kelas diagram *increment* 1 adalah kelas Point, Path, dan PointOfInterest yang berfungsi sebagai Model untuk data titik, jalur, dan titik tujuan. Kemudian juga terdapat kelas DatabaseHelper untuk melakukan perubahan data pada *database* dan DatabaseProvider untuk menyediakan data yang diminta melalui *query* yang dimasukkan.

5.1.4 Perancangan Algoritma *Increment* 1

Perancangan algoritma merupakan perancangan yang menjelaskan algoritma untuk menjalankan fungsionalitas sistem. Algoritma ini yang akan menjadi dasar dalam pembuatan kode pada tahap implementasi. Terdapat 3 algoritma yang dijabarkan pada perancangan algoritma *increment* 1, pemilihan 3 algoritma ini berdasarkan fungsi yang sering dilakukan dalam pemakaian aplikasi.

5.1.4.1 Perancangan Algoritma Menampilkan Titik Dan Jalur Dari Graf Yang DIpilih

Algoritma Menampilkan Titik Dan Jalur Dari Graf Yang DIpilih digunakan untuk menampilkan titik dan jalur dari graf yang dipilih. Algoritma ini menghasilkan *background task* yang digunakan untuk mengambil Points dan Paths dari *database*. Algoritma Menampilkan Titik Dan Jalur Dari Graf Yang DIpilih dijelaskan pada Tabel 5.3.

Tabel 5.3 Penjelasan Algoritma Menampilkan Titik Dan Jalur Dari Graf Yang DIpilih

Nama Kelas	:	MapsActivity
Nama Metode	:	onMapReady()
Deskripsi	:	Method ini merupakan method yang dijalankan ketika tampilan peta pada halaman mengubah graf telah siap untuk menerima interaksi dari pengguna. Method ini berfungsi untuk menyiapkan penangkap interaksi pada peta dan memicu *background task* untuk mengambil data titik dan jalur pada *database*.
Algoritma	:	

```
Mulai
Atur nilai Google Map
Atur penangkap interaksi tap pada tampilan peta
```

```
Atur penangkap interaksi tap Marker
Mulai background task untuk mengambil data titik dan jalur
Selesai
```

5.1.4.2 Perancangan Algoritma Menambah Titik Dalam Graf

Algoritma Menambah Titik Dalam Graf digunakan untuk memasukkan titik baru ke dalam *database*. Algoritma ini membuat kelas *concrete command* menambah titik dan menggunakan kelas *invoker* untuk menjalankan fungsi pada kelas *concrete command* yang dibuat. Algoritma Menambah Titik Dalam Graf dijelaskan pada Tabel 5.4.

Tabel 5.4 Penjelasan Algoritma Menambah Titik Dalam Graf

Nama Kelas	:	MapsActivity
Nama Metode	:	onMapClikListener()
Deskripsi	:	Metode ini merupakan method yang dijalankan ketika pengguna melakukan *tap* pada tampilan peta. Dalam aplikasi ini, metode akan menjalankan fungsionalitas untuk mengubah data graf dengan menambah titik dalam graf pada lokasi peta yang di *tap* oleh pengguna.
Algoritma	:	

```
Mulai
Ambil nilai Latitude dan Longitude lokasi yang dipilih
Buat instans kelas Concrete Command untuk menambah titik
Panggil kelas Invoker untuk menjalankan fungsi pada Concrete
Command yang dibuat
Selesai
```

5.1.4.3 Perancangan Algoritma Menghapus Titik

Algoritma Menghapus Titik digunakan untuk menghapus suatu titik dari *database*. Algoritma ini membuat kelas *concrete command* untuk menghapus titik dan menggunakan kelas *invoker* untuk menjalankan fungsi pada kelas *concrete command* yang dibuat. Algoritma menghapus titik dijelaskan pada Tabel 5.5.

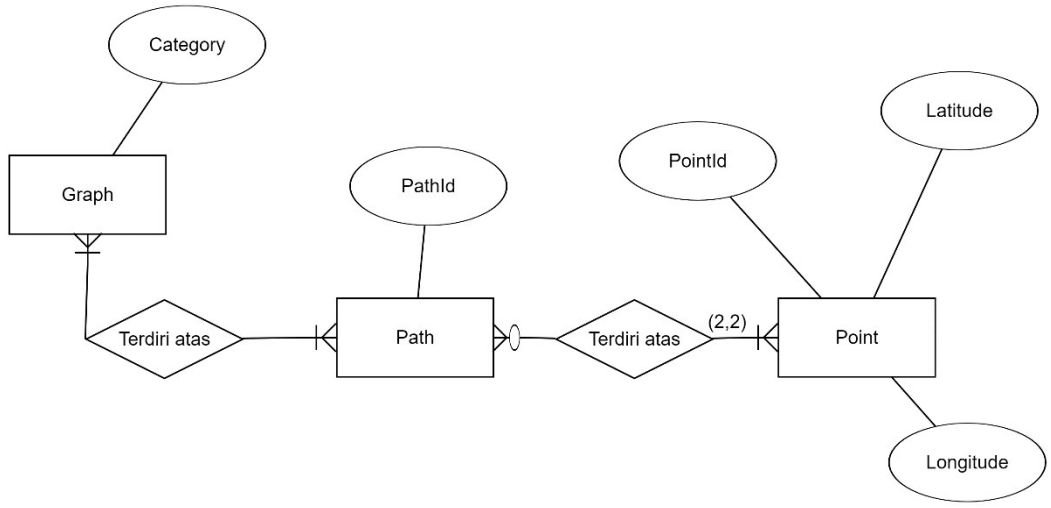
Tabel 5.5 Penjelasan Algoritma Menghapus Titik

Nama Kelas	:	MapsActivity
Nama Metode	:	deletePoint()
Deskripsi	:	Metode ini merupakan method yang dijalankan ketika pengguna melakukan *tap* pada tombol hapus. Dalam

	aplikasi ini, metode akan menjalankan fungsionalitas untuk menghapus titik yang telah dipilih oleh pengguna.
Algoritma :	

```
Mulai
Ambil objek Point yang akan dihapus
Buat instans kelas Concrete Command untuk menghapus titik
Panggil kelas Invoker untuk menjalankan fungsi pada Concrete
Command yang dibuat
Selesai
```

5.1.5 Perancangan Relasional Data Model *Increment* 1



Gambar 5.11 Pemodelan Relasional Data Model *Increment* 1

Terdapat 3 Entitas yang ditemukan pada Pemodelan Relasional Data Model *Increment* 1, yaitu Graph, Path, dan Point. Hubungan relasi yang terjadi di antara 3 entitas tersebut adalah Graph dan Path, dimana Sebuah Graph terdiri atas satu atau banyak Path (*One to (One or Many)*). Kemudian terdapat relasi antara Path dan Point, dimana sebuah Path terdiri atas satu atau banyak Point dengan nilai minimum 2 dan maksimal 2 (*One to (One or Many)*).

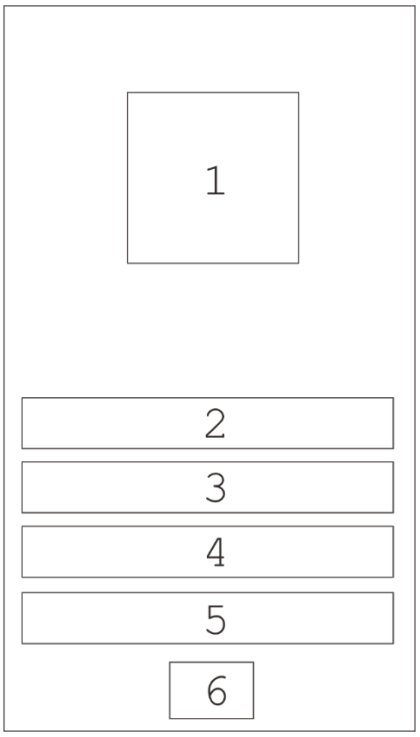
5.1.6 Perancangan Antarmuka *Increment* 1

Aplikasi Mobile Struktur Data Graf menggunakan perangkat Android sebagai antarmuka pengguna untuk melakukan interaksi di dalamnya. Dengan menggunakan perangkat Android, maka pengguna dapat berinteraksi menggunakan layar sentuh dan virtual keyboard untuk mengoperasikan aplikasi.

Rancangan antarmuka dari Aplikasi Mobile Struktur Data Graf adalah sebagai berikut.

5.1.6.1 Perancangan Antarmuka Halaman Login

Halaman login diakses oleh pengguna untuk memasukkan data email dan password sebagai syarat autentikasi. Halaman ini juga akan menampilkan pesan *error* jika terdapat kesalahan dalam proses autentikasi.



Gambar 5.12 Rancangan Antarmuka Halaman Login *Increment* 1

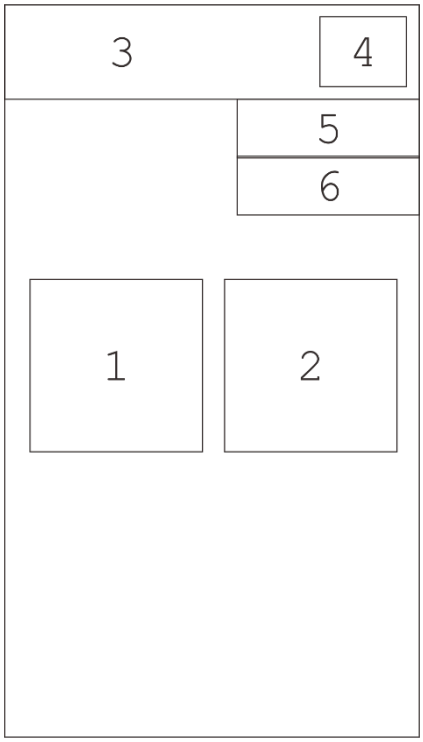
Tabel 5.6 Penjelasan Rancangan Antarmuka Halaman Login *Increment* 1

No	Nama Objek	Tipe	Keterangan
1	Application Logo	ImageView	ImageView untuk menampilkan logo aplikasi
2	Email Form	EditText	Untuk memasukkan alamat email milik pengguna
3	Password Form	EditText	Untuk memasukkan password akun milik pengguna

4	Submit Button	Button	Untuk melakukan proses *login*
5	Error Message	TextView	Untuk menampilkan pesan kesalahan yang terjadi ketika proses *login* dilakukan
6	Loading ProgressBar	ProgressBar	Untuk menampilkan ProgressBar ketika proses *login* sedang dilakukan

5.1.6.2 Perancangan Antarmuka Halaman Utama

Halaman utama diakses oleh pengguna ketika pengguna telah berhasil *login* ke dalam aplikasi. Pada halaman ini, terdapat pilihan edit graf berdasarkan tipe nya, yaitu graf pejalan kaki dan graf kendaraan bermotor. Selain itu, juga terdapat pilihan untuk *logout* dan untuk mengakses menu *setting* pada bagian menu.



Gambar 5.13 Rancangan Antarmuka Halaman Utama *Increment* 1

Tabel 5.7 Penjelasan Rancangan Antarmuka Halaman Utama *Increment* 1

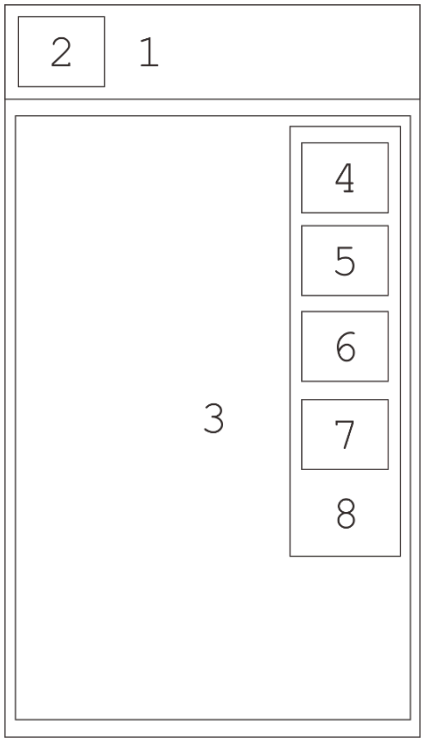
No	Nama Objek	Tipe	Keterangan
1	Opsi Edit Graf Pejalan Kaki	ImageView	Untuk mengakses halaman edit graf pejalan kaki
2	Opsi Edit Graf Kendaraan Bermotor	ImageView	Untuk mengakses halaman edit graf kendaraan bermotor
3	Action Bar	ActionBar	Untuk menampilkan nama aplikasi
4	Menu Button	MenuButton	Untuk mengakses menu item yang tersedia
5	Setting Item	MenuItem	Untuk mengakses halaman *setting*
6	Logout Item	MenuItem	Untuk keluar dari akun yang sedang *login*

5.1.6.3 Perancangan Antarmuka Halaman Edit Graf

Halaman edit graf di *reuse* untuk menampilkan halaman edit graf untuk semua tipe graf. Oleh karena itu, tipe graf pejalan kaki dan tipe graf kendaraan bermotor menggunakan *resource* yang sama untuk menampilkan halaman edit graf. Pada halaman edit graf, terdapat Fragment peta yang memunculkan peta Google Maps. Di dalam Fragment peta ini terdapat Marker dan Polyline yang membentuk graf.

Pengguna dapat melakukan *tap* di area Fragment peta untuk memasukkan titik baru. Pengguna juga dapat melakukan *pinch* pada tampilan peta untuk melakukan perbesaran dan pengecilan tampilan peta. Selain itu, pengguna juga dapat melakukan *tap* pada Marker untuk memilih Marker tersebut.

Di sisi sebelah kanan terdapat panel tombol sebagai pendukung bagi pengguna ketika sedang melakukan proses mengedit Graf. Panel tombol berisi tombol-tombol pendukung yaitu tombol *undo*¸ tombol *redo*, tombol hapus, dan tombol *reset*.



Gambar 5.14 Rancangan Antarmuka Halaman Edit Graf *Increment* 1

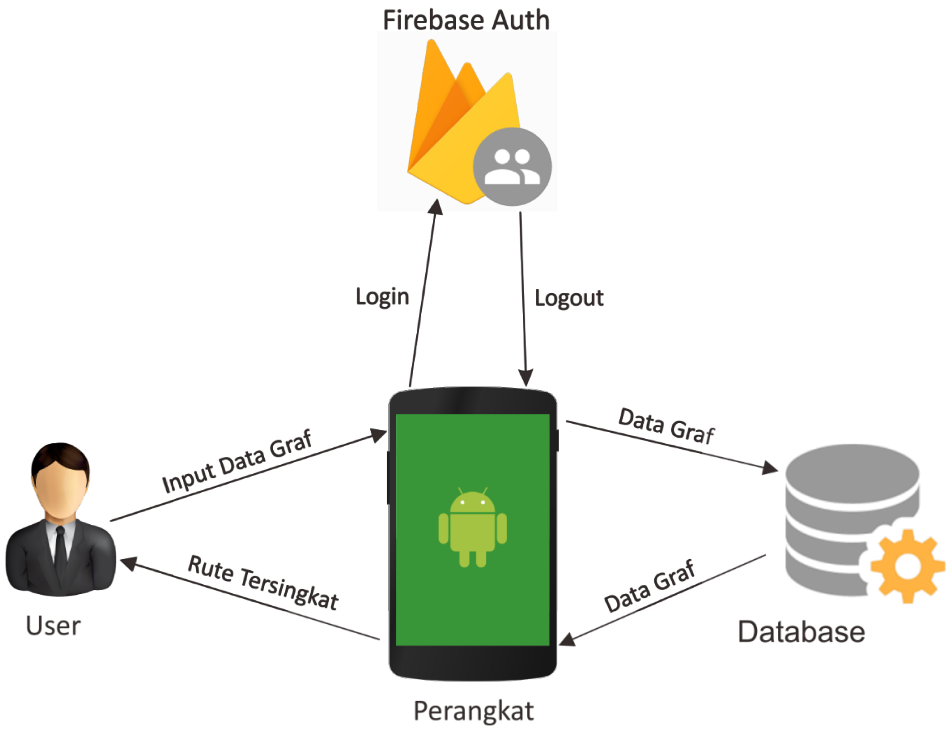
Tabel 5.8 Penjelasan Rancangan Antarmuka Halaman Edit Graf *Increment* 1

No	Nama Objek	Tipe	Keterangan
1	Action Bar	ActionBar	Untuk menampilkan nama graf yang sedang di edit
2	Back Button	MenuButton	Untuk kembali ke halaman utama
3	Map Fragment	Fragment	Untuk menampilkan tampilan peta Google Maps
4	Undo Button	FloatingActionButton	Untuk melakukan aksi *undo*
5	Redo Button	FloatingActionButton	Untuk melakukan aksi *redo*
6	Delete Button	FloatingActionButton	Untuk menghapus

			suatu titik pada graf
7	Reset Button	FloatingActionButton	Untuk melakukan aksi *reset*, yaitu menghapus semua titik dan jalur yang terdapat suatu tipe graf
8	Panel Button	LinearLayout	Sebagai *panel* atau tempat untuk menampung tombol-tombol pendukung dalam melakukan *edit* graf

5.1.7 Perancangan Arsitektural *Increment* 2

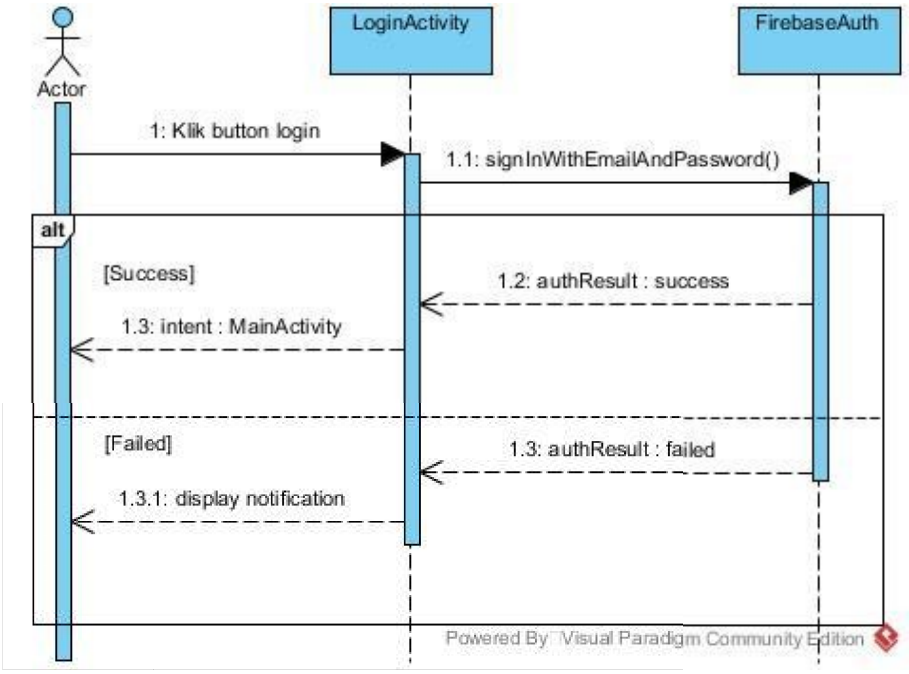
Sistem Aplikasi Mobile Struktur Data Graf berjalan pada perangkat mobile dengan sistem operasi Android. Pengguna harus terautentikasi terlebih dahulu untuk menjalankan fungsionalitas aplikasi. Proses autentikasi pengguna dilakukan menggunakan backend service Firebase Authentication yang dapat mengautentikasi pengguna menggunakan email dan password yang telah didaftarkan sebelumnya. Autentikasi yang berhasil nantinya akan membuka akses fungsionalitas aplikasi kepada pengguna. Pengguna berinteraksi dengan perangkat untuk memasukkan data graf yang terdiri atas titik-titik yang saling terhubung membentuk jalur. Data graf ini akan disimpan dalam database lokal Android dengan menggunakan sistem basis data SQLite dan hanya bisa diakses oleh pengguna yang telah terautentikasi. Data graf yang disimpan ini nantinya akan diuji dengan mengimplementasikannya menggunakan algoritma Djikstra untuk mendapatkan rute tersingkat dari suatu lokasi ke salah satu lokasi di Universitas Brawijaya. Perancangan Arsitektural Sistem pada *increment* 2 dapat dilihat pada Gambar 5.15.



Gambar 5.15 Perancangan Arsitektural *Increment* 2

5.1.8 Perancangan *Sequence* Diagram *Increment* 2

5.1.8.1 *Sequence* Diagram Login

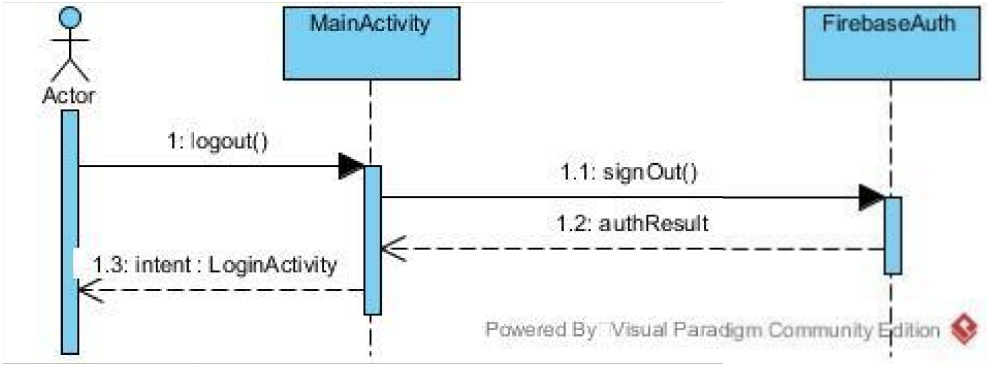


Gambar 5.16 *Sequence* Diagram Login *Increment* 2

Pada Gambar 5.16, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Login.

Pada *Sequence* Diagram Login, aktor memasukkan data berupa *email* dan *password*. Setelah data diisi, aktor menekan tombol *login* untuk melakukan proses *login*. Kemudian sistem akan memanggil method signInWithEmailAndPassword() dari kelas FirebaseAuth untuk mengautentikasi user ke dalam aplikasi. Jika kembalian dari FirebaseAuth yaitu authResult bernilai *success*, maka aktor akan mendapatkan *intent* yang akan mengubah tampilan ke halaman MainActivity untuk membuka akses fungsionalitas aplikasi kepada pengguna. Jika kembalian dari FirebaseAuth yaitu authResult bernilai *failed*, maka aktor akan mendapatkan *intent* yang akan memunculkan notifikasi kesalahan.

5.1.8.2 *Sequence* Diagram Logout

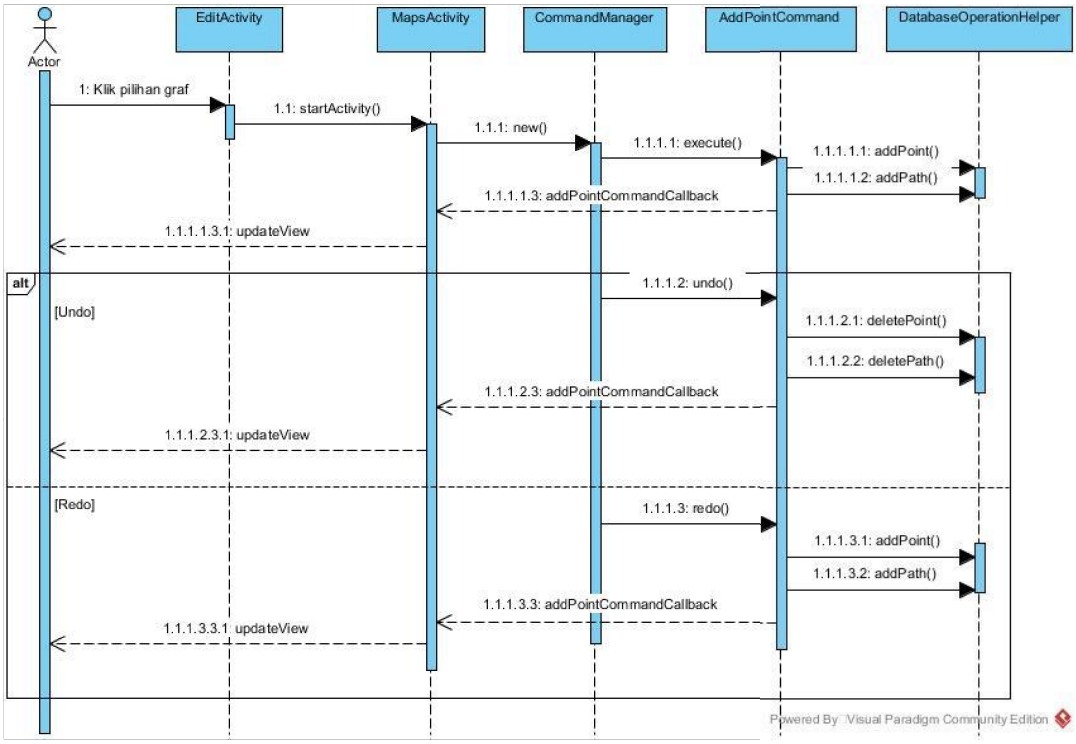


Gambar 5.17 *Sequence* Diagram Logout *Increment* 2

Pada Gambar 5.17, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Logout.

Pada *Sequence* Diagram Logout, aktor menekan tombol *logout* untuk keluar dari aplikasi. Setelah itu, sistem akan memanggil method signOut() dari kelas FirebaseAuth untuk mengeluarkan aktor dari sistem. Setelah itu, aktor akan dipindahkan ke halaman LoginActivity.

5.1.8.3 *Sequence* Diagram Membuat Titik Dalam Graf

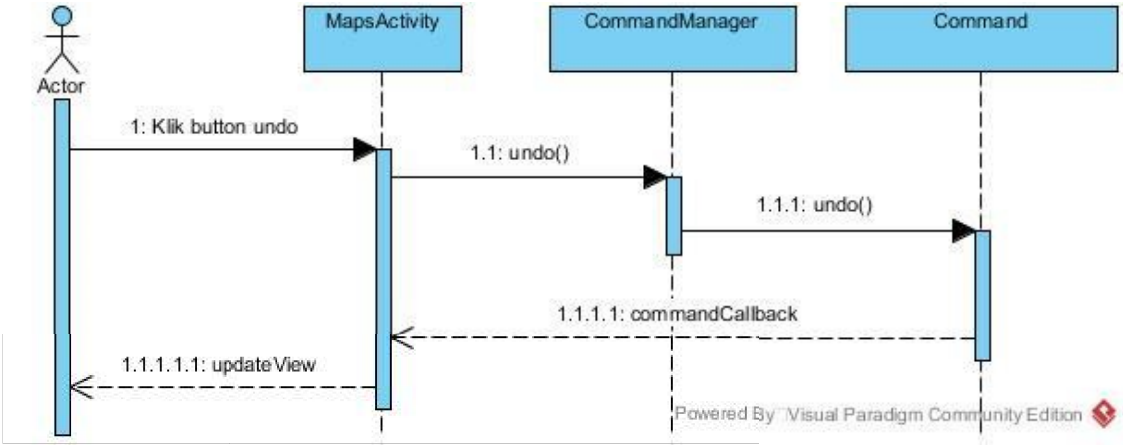


Gambar 5.18 *Sequence* Diagram Membuat Titik Dalam Graf *Increment* 2

Pada Gambar 5.18, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Membuat Titik Dalam Graf.

Pada *Sequence* Diagram Membuat Titik Dalam Graf, pengguna memilih terlebih dahulu graf yang akan di edit pada EditActivity. Setelah itu, sistem akan memanggil MapsActivity beserta variable pilihan data yang dipilih oleh aktor. Fungsionalitas mengedit graf menggunakan *Command Pattern* dalam menjalankan fungsi-fungsi nya. Pada konteks ini, kelas MapsActivity bertindak sebagai *client*¸ kelas CommandManager bertindak sebagai *invoker*, kelas DatabaseOperationHelper bertindak sebagai *receiver*, dan kelas AddPointCommand bertindak sebagai *Concrete Command* yang merupakan *child* dari kelas Command. MapsActivity akan membuat instans CommandManager, setelah itu instans CommandManager ini akan digunakan untuk memanggil method execute, undo, dan redo pada setiap Command yang terdapat pada stack di dalam CommandManager. Aksi *execute* akan membuat *point* dan *path* baru pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi database. Aksi *undo* akan menghapus *point* dan *path* yang telah ditambahkan pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Aksi *redo* akan memunculkan kembali *point* dan *path* yang terhapus ketika aksi *undo* dijalankan dengan membuat *point* dan *path* baru dengan menggunakan kelas DabaseOperationHelper sesuai dengan data yang ada pada aksi *execute*. Setiap aksi yang dilakukan (*execute*, *undo*, dan *redo*) akan memicu *callback* addPointCommandCallback yang akan melakukan pembaruan terhadap perubahan yang terjadi ke tampilan peta.

5.1.8.4 *Sequence* Diagram Undo



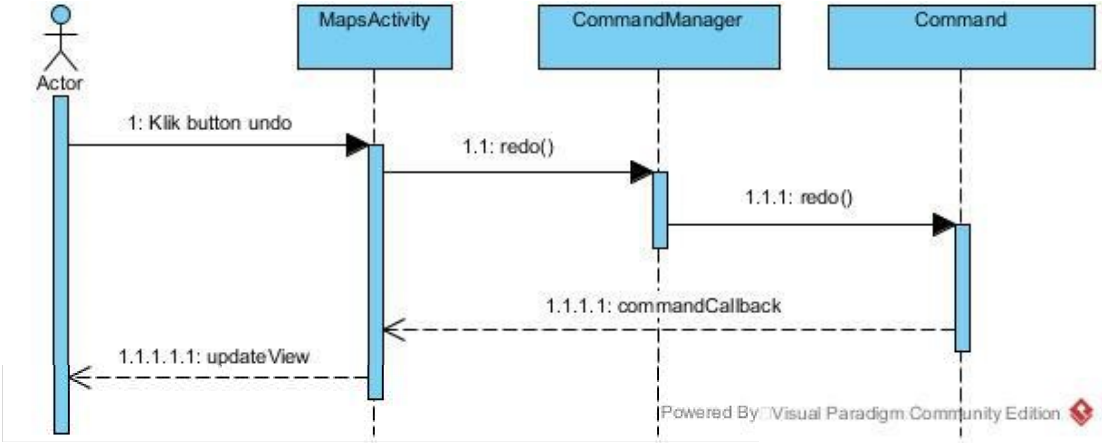
Gambar 5.19 Sequence Diagram Undo

Pada Gambar 5.4, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Undo.

Pada *Sequence* Diagram Undo, pengguna melakukan *tap* pada tombol *undo* untuk mengembalikan kondisi graf ke aksi sebelumnya. Tombol *undo* akan memanggil kelas CommandManager sebagai kelas *invoker* untuk melakukan fungsi *undo*. Kemudian, CommandManager akan memicu Command pada *stack* teratas nya untuk menjalankan fungsi *undo*. Selanjutnya, Command yang

melakukan fungsi *undo* akan mengembalikan *callback* ke MapsActivity untuk melakukan *update* terhadap *view* pengguna.

5.1.8.5 *Sequence* Diagram Redo

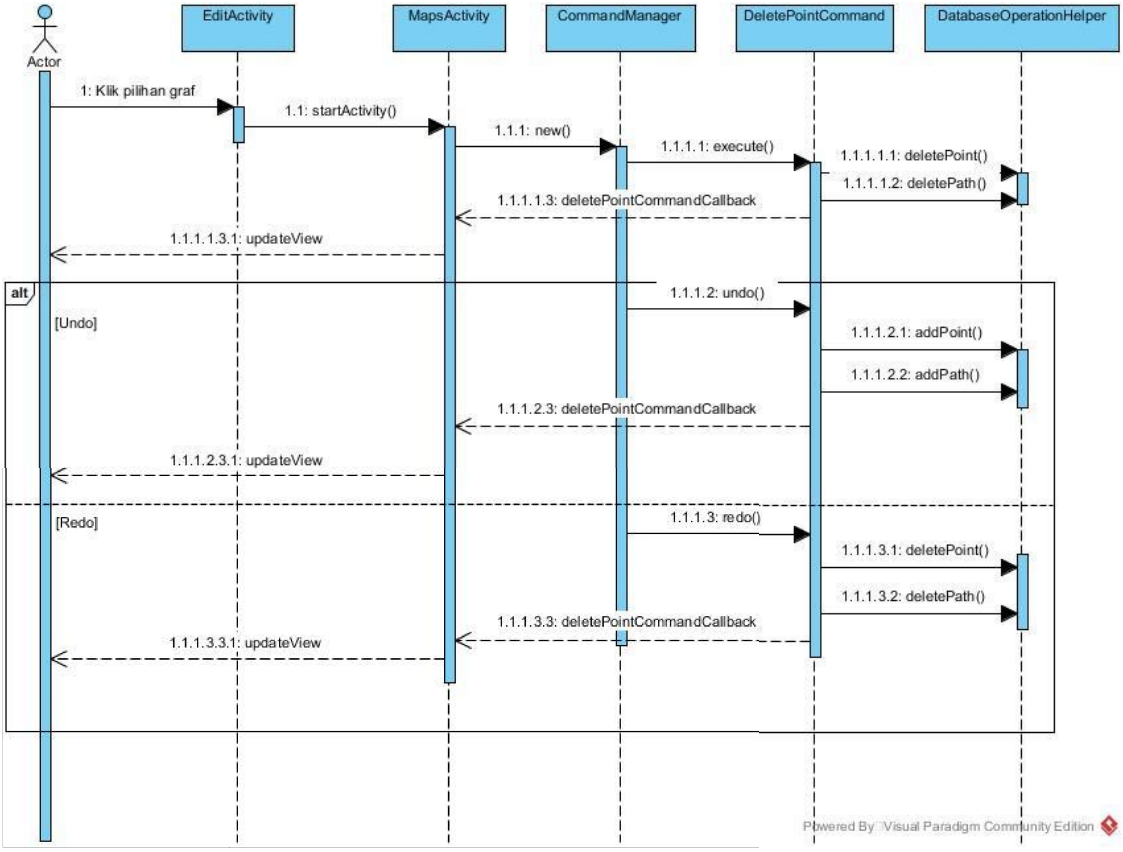


Gambar 5.20 Sequence Diagram Redo

Pada Gambar 5.4, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Redo.

Pada *Sequence* Diagram Undo, pengguna melakukan *tap* pada tombol *redo* untuk mengembalikan kondisi graf ke aksi sebelum aksi *undo* dilakukan. Tombol *redo* akan memanggil kelas CommandManager sebagai kelas *invoker* untuk melakukan fungsi *redo*. Kemudian, CommandManager akan memicu Command pada *stack* teratas nya untuk menjalankan fungsi *redo*. Selanjutnya, Command yang melakukan fungsi *redo* akan mengembalikan *callback* ke MapsActivity untuk melakukan *update* terhadap *view* pengguna.

5.1.8.6 *Sequence* Diagram Menghapus Titik



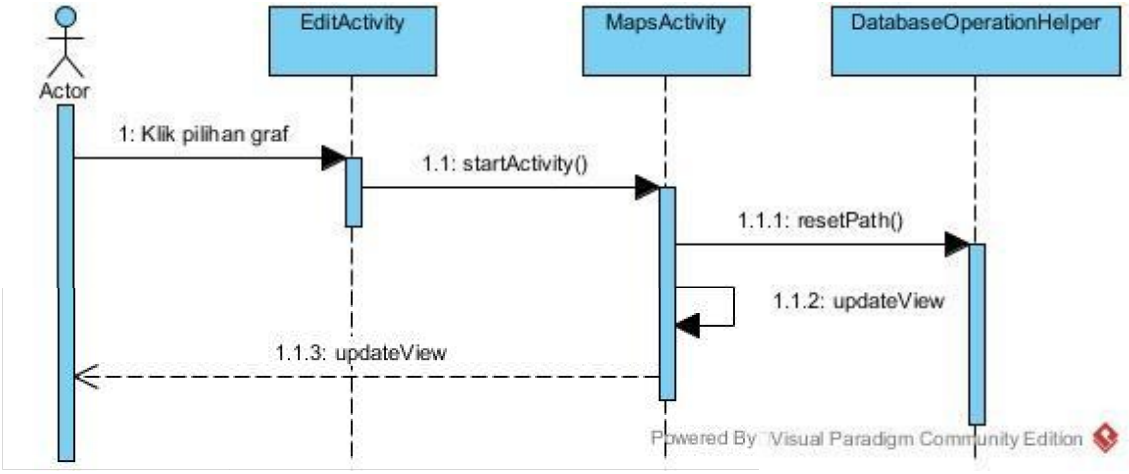
Gambar 5.21 Sequence Diagram Menghapus Titik *Increment* 2

Pada Gambar 5.21, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Menghapus Titik.

Pada *Sequence* Diagram Menghapus Titik, pengguna memilih terlebih dahulu graf yang akan di edit pada EditActivity. Setelah itu, sistem akan memanggil MapsActivity beserta variable pilihan data yang dipilih oleh aktor. Fungsionalitas mengedit graf menggunakan *Command Pattern* dalam menjalankan fungsi-fungsi nya. Pada konteks ini, kelas MapsActivity bertindak sebagai *client*¸ kelas CommandManager bertindak sebagai *invoker*, kelas DatabaseOperationHelper bertindak sebagai *receiver*, dan kelas DeletePointCommand bertindak sebagai *Concrete Command* yang merupakan *child* dari kelas Command. MapsActivity akan membuat instans CommandManager, setelah itu instans CommandManager ini akan digunakan untuk memanggil method *execute*, *undo*, dan *redo* pada setiap Command yang terdapat pada stack di dalam CommandManager. Aksi *execute* akan menghapus *point* dan *path* yang ditambahkan pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Aksi *undo* akan membuat kembali *point* dan *path* yang telah terhapus ketika aksi *execute* dijalankan dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi database. Aksi *redo* akan menghapus kembali *point* dan *path* yang telah dikembalikan ketika aksi *undo* dijalankan dengan menggunakan kelas

DabaseOperationHelper untuk melakukan operasi *database*. Setiap aksi yang dilakukan (*execute*, *undo*, dan *redo*) akan memicu *callback* addPointCommandCallback yang akan melakukan pembaruan terhadap perubahan yang terjadi ke tampilan peta.

5.1.8.7 *Sequence* Diagram Reset Graf

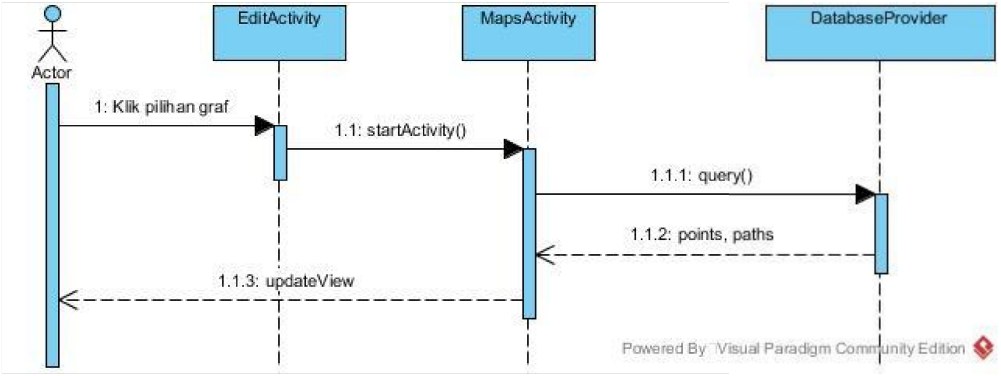


Gambar 5.22 Sequence Diagram Reset Graf *Increment* 2

Pada Gambar 5.22, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Reset Graf.

Pada *Sequence* Diagram Reset Graf, aktor memilih terlebih dahulu jenis graf yang akan di reset. Setelah itu sistem akan menampilkan halaman MapsActivity. Pada MapsActivity, terdapat method yang akan memanggil method resetPath() pada kelas DatabaseOperationHelper yang akan menghapus seluruh titik dan jalur pada graf yang dipilih. Setelah itu, MapsActivity akan melakukan updateView untuk menghilangkan jalur dan titik yang ada pada tampilan peta.

5.1.8.8 *Sequence* Diagram Melihat Graf

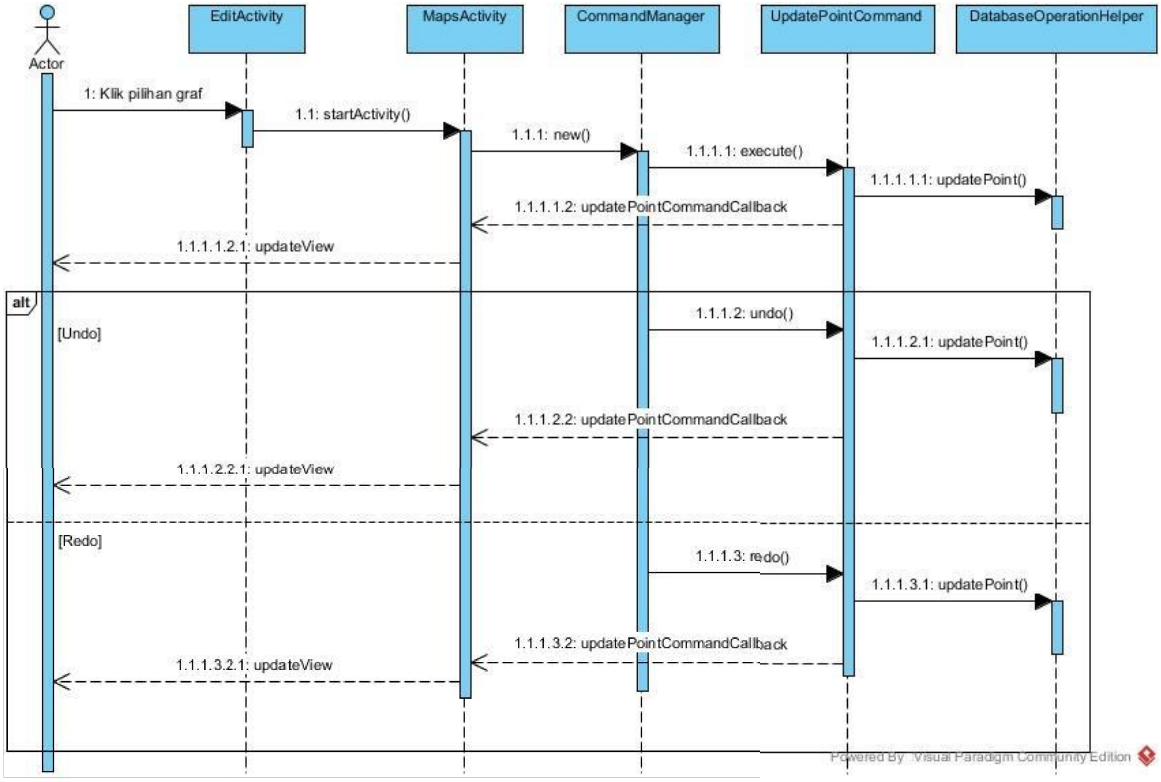


Gambar 5.23 Sequence Diagram Melihat Graf *Increment* 2

Pada Gambar 5.23, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Melihat Graf.

Pada *Sequence* Diagram Logout, aktor memilih terlebih dahulu jenis graf yang akan di *reset*. Setelah itu sistem akan menampilkan halaman MapsActivity. Pada MapsActivity, terdapat *method* yang akan memanggil method query() untuk melakukan pengambilan data dari *database* menggunakan DatabaseProvider. DatabaseProvider akan memberikan data graf yang diminta, selanjutnya MapsActivity merubah *view* nya dengan menambah *marker* dan *polyline* sesuai data graf yang didapatkan.

5.1.8.9 *Sequence* Diagram Mengubah Lokasi Titik



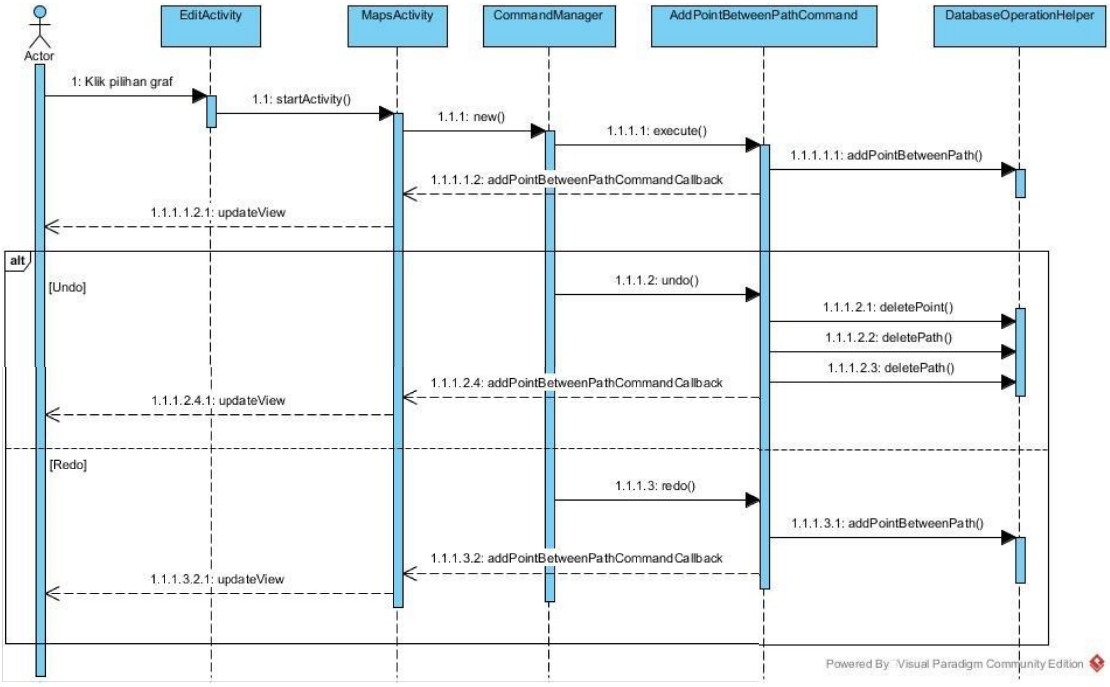
Gambar 5.24 Sequence Diagram Mengubah Lokasi Titik *Increment* 2

Pada Gambar 5.24, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Mengubah Lokasi Titik.

Pada *Sequence* Diagram Mengubah Lokasi Titik, pengguna memilih terlebih dahulu graf yang akan di edit pada EditActivity. Setelah itu, sistem akan memanggil MapsActivity beserta variable pilihan data yang dipilih oleh aktor. Fungsionalitas mengedit graf menggunakan *Command Pattern* dalam menjalankan fungsi-fungsi nya. Pada konteks ini, kelas MapsActivity bertindak sebagai *client*¸ kelas CommandManager bertindak sebagai *receiver*, dan kelas UpdatePointCommand bertindak sebagai *Concrete Command* yang merupakan *child* dari kelas Command. MapsActivity akan membuat instans CommandManager, setelah itu instans CommandManager ini akan digunakan untuk memanggil method execute, undo, dan redo pada setiap Command yang terdapat pada stack di dalam CommandManager. Aksi *execute* akan melakukan update terhadap *point* yang diubah pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi database. Aksi *undo* akan mengembalikan posisi *point* yang telah di update pada database pada aksi *execute* dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Aksi *redo* akan melakukan *update* kembali terhadap *point* yang telah dikembalikan ketika aksi *undo* dijalankan dengan menggunakan kelas DabaseOperationHelper sesuai dengan data yang ada pada aksi *execute*.Setiap aksi yang dilakukan (*execute*, *undo*,

dan *redo*) akan memicu *callback* updatePointCommandCallback yang akan melakukan pembaruan terhadap perubahan yang terjadi ke tampilan peta.

5.1.8.10 *Sequence Diagram* Membuat Titik Diantara 2 Titik Dalam 1 Jalur



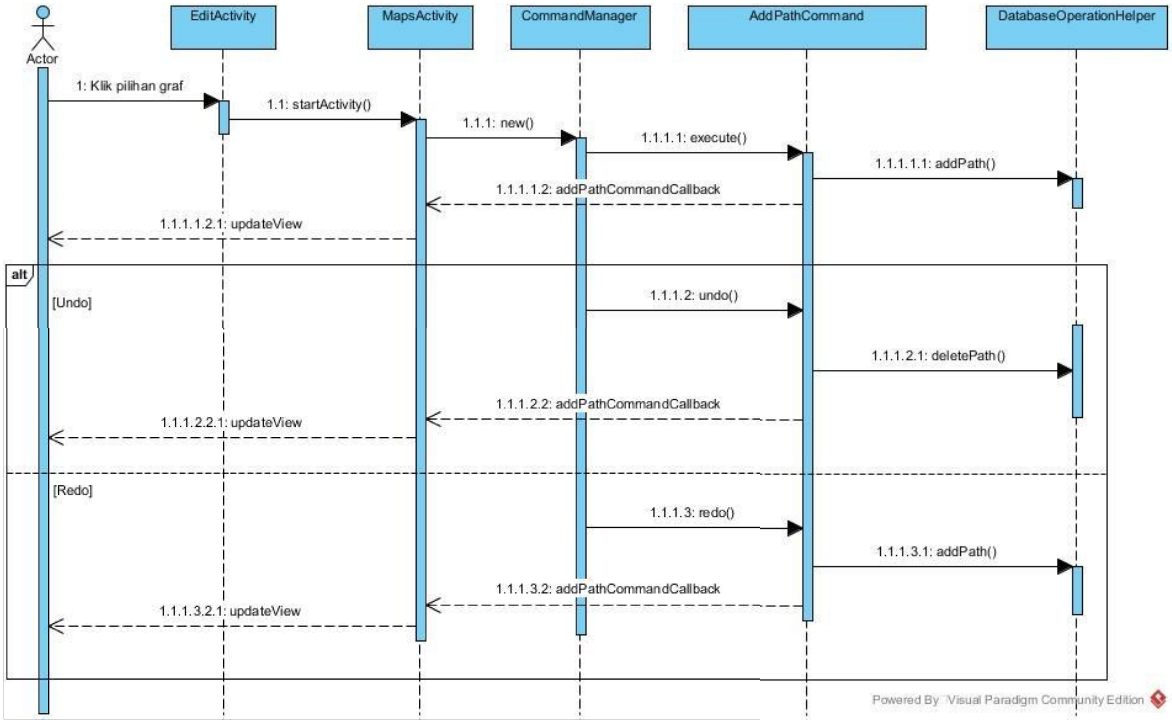
Gambar 5.25 Sequence Diagram Membuat Titik Diantara 2 Titik Dalam 1 Jalur *Increment* 2

Pada Gambar 5.25, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Membuat Titik Diantara 2 Titik Dalam 1 Jalur.

Pada *Sequence* Diagram Membuat Titik Diantara 2 Titik Dalam 1 Jalur, pengguna memilih terlebih dahulu graf yang akan di edit pada EditActivity. Setelah itu, sistem akan memanggil MapsActivity beserta variable pilihan data yang dipilih oleh aktor. Fungsionalitas mengedit graf menggunakan *Command Pattern* dalam menjalankan fungsi-fungsi nya. Pada konteks ini, kelas MapsActivity bertindak sebagai *client*, kelas CommandManager bertindak sebagai *receiver*, dan kelas AddPathBetweenPointCommand bertindak sebagai *Concrete Command* yang merupakan *child* dari kelas Command. MapsActivity akan membuat instans CommandManager, setelah itu instans CommandManager ini akan digunakan untuk memanggil method *execute*, *undo*, dan *redo* pada setiap Command yang terdapat pada stack di dalam CommandManager. Aksi *execute* akan menambahkan *point* diantara 2 titik dalam 1 jalur dengan membuat 1 *point* baru dan 2 *path* baru pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Aksi *undo* akan menghapus 1 *point* dan 2 *path* yang dibuat ketika aksi *execute* dijalankan dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi database. Aksi *redo* akan menambahkan kembali 1 *point* dan 2 *path* yang telah dihapus ketika aksi *undo* dijalankan dengan menggunakan kelas DabaseOperationHelper

untuk melakukan operasi *database*. Setiap aksi yang dilakukan (*execute*, *undo*, dan *redo*) akan memicu *callback* addPathBetweenPathCommandCallback yang akan melakukan pembaruan terhadap perubahan yang terjadi ke tampilan peta.

5.1.8.11 *Sequence Diagram* Menyatukan 2 Titik



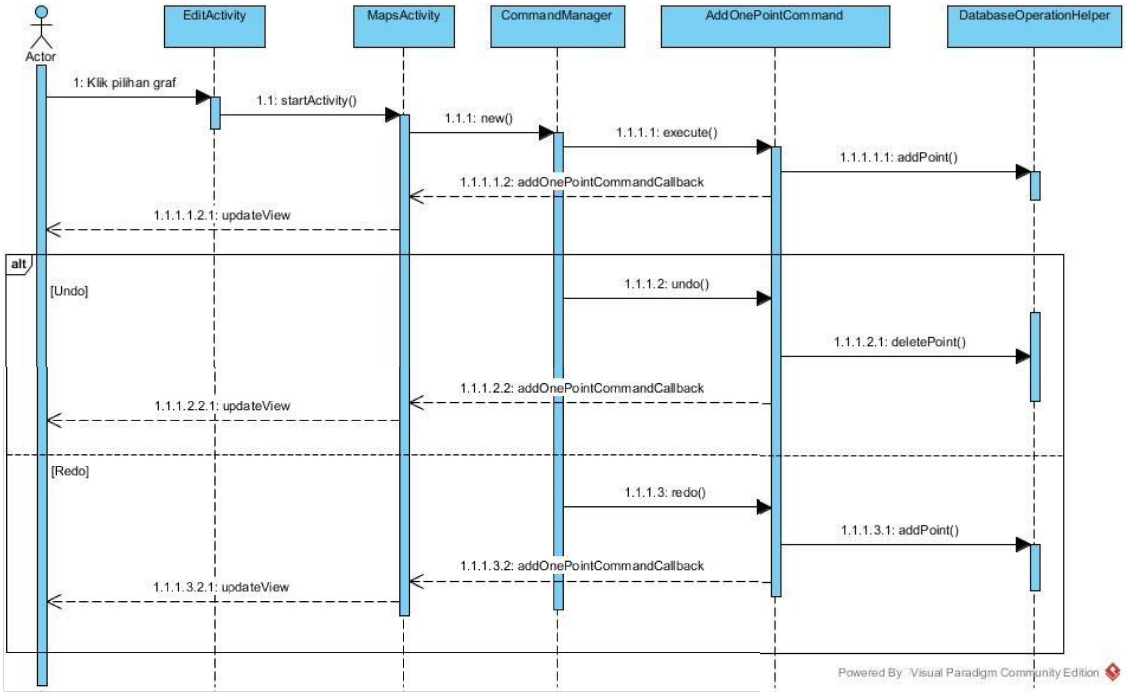
Gambar 5.26 Sequence Diagram Menyatukan 2 Titik *Increment* 2

Pada Gambar 5.26, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Menyatukan 2 Titik.

Pada *Sequence* Diagram Menyatukan 2 Titik, pengguna memilih terlebih dahulu graf yang akan di edit pada EditActivity. Setelah itu, sistem akan memanggil MapsActivity beserta variable pilihan data yang dipilih oleh aktor. Fungsionalitas mengedit graf menggunakan *Command Pattern* dalam menjalankan fungsi-fungsi nya. Pada konteks ini, kelas MapsActivity bertindak sebagai *client*¸ kelas CommandManager bertindak sebagai *receiver*, dan kelas AddPathCommand bertindak sebagai *Concrete Command* yang merupakan *child* dari kelas Command. MapsActivity akan membuat instans CommandManager, setelah itu instans CommandManager ini akan digunakan untuk memanggil method *execute*, *undo*, dan *redo* pada setiap Command yang terdapat pada stack di dalam CommandManager. Aksi *execute* akan menambahkan *path* baru pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Aksi *undo* akan menghapus *path* yang ditambahkan ketika aksi *execute* dijalankan dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi database. Aksi *redo* akan menambahkan kembali *path* yang telah terhapus ketika aksi *undo* dijalankan dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Setiap aksi yang dilakukan (*execute*, *undo*, dan

redo) akan memicu *callback* addPathCommandCallback yang akan melakukan pembaruan terhadap perubahan yang terjadi ke tampilan peta.

5.1.8.12 *Sequence Diagram* Membuat Titik Diluar Graf



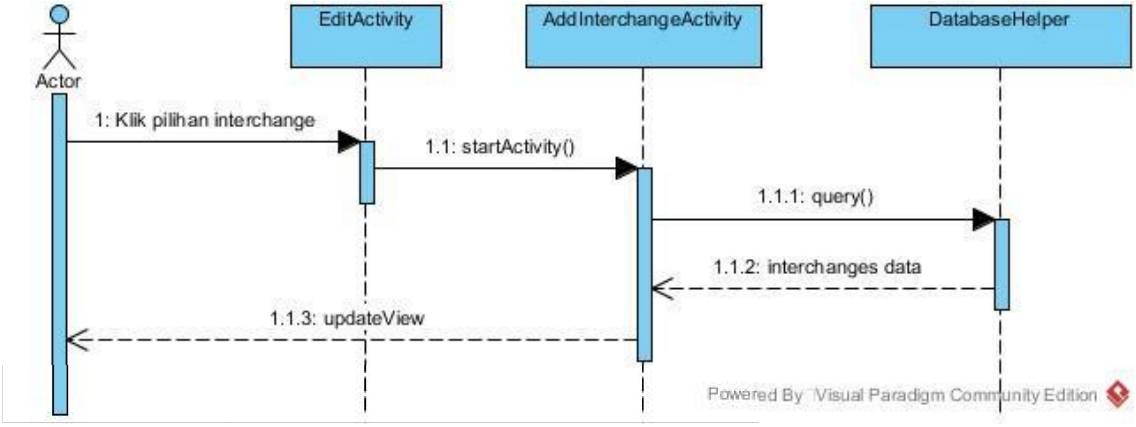
Gambar 5.27 Sequence Diagram Membuat Titik Diluar Graf *Increment* 2

Pada Gambar 5.27, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Membuat Titik Diluar Graf.

Pada *Sequence* Diagram Reset Graf, pengguna memilih terlebih dahulu graf yang akan di edit pada EditActivity. Setelah itu, sistem akan memanggil MapsActivity beserta variable pilihan data yang dipilih oleh aktor. Fungsionalitas mengedit graf menggunakan *Command Pattern* dalam menjalankan fungsi-fungsi nya. Pada konteks ini, kelas MapsActivity bertindak sebagai *client*¸ kelas CommandManager bertindak sebagai *receiver*, dan kelas AddOnePointCommand bertindak sebagai *Concrete Command* yang merupakan *child* dari kelas Command. MapsActivity akan membuat instans CommandManager, setelah itu instans CommandManager ini akan digunakan untuk memanggil method *execute*, *undo*, dan *redo* pada setiap Command yang terdapat pada stack di dalam CommandManager. Aksi *execute* akan menambahkan *point* baru pada database dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Aksi *undo* akan menghapus *point* yang ditambahkan ketika aksi *execute* dijalankan dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi database. Aksi *redo* akan menambahkan kembali *point* yang telah terhapus ketika aksi *undo* dijalankan dengan menggunakan kelas DabaseOperationHelper untuk melakukan operasi *database*. Setiap aksi yang dilakukan (*execute*, *undo*, dan *redo*) akan memicu *callback*

addOnePointCommandCallback yang akan melakukan pembaruan terhadap perubahan yang terjadi ke tampilan peta.

5.1.8.13 *Sequence Diagram* Melihat *Interchange*

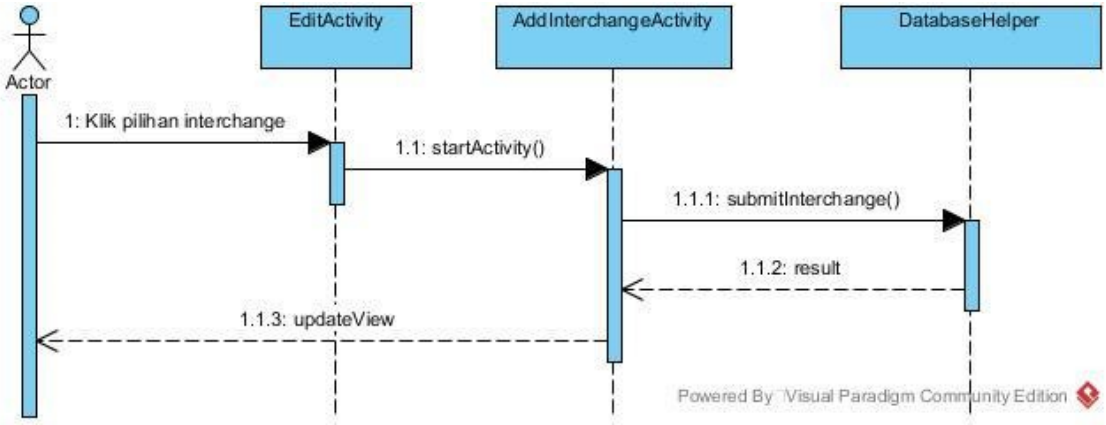


Gambar 5.28 Sequence Diagram Melihat Interchange *Increment* 2

Pada Gambar 5.28, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Melihat Interchange.

Pada *Sequence* Diagram Melihat Interchange, aktor memilih terlebih dahulu pilihan *edit interchange* melalui EditActivity. Setelah itu sistem akan menampilkan halaman AddInterchangeActivity. Pada kelas AddInterchangeActivity, sistem akan mengambil data *interchange* pada *database* dengan menggunakan kelas DatabaseHelper. Setelah mendapatkan data *interchange* dari *database*, sistem akan menampilkan lokasi-lokasi *interchange* yang ada pada tampilan peta dalam bentuk Marker.

5.1.8.14 *Sequence Diagram* Memasukkan *Interchange*

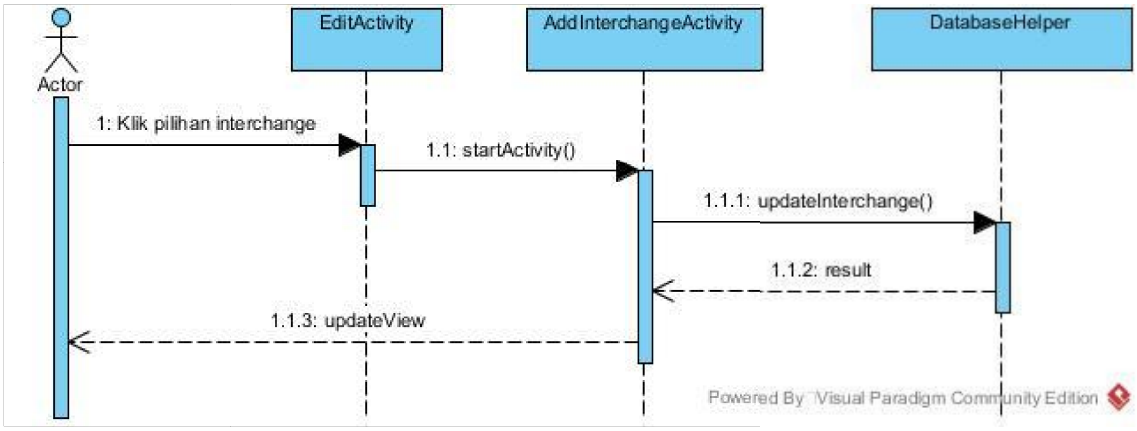


Gambar 5.29 Sequence Diagram Memasukkan *Interchange Increment* 2

Pada Gambar 5.29, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Memasukkan Interchange.

Pada *Sequence* Diagram Memasukkan Interchange, aktor memilih terlebih dahulu pilihan *edit interchange* melalui EditActivity. Setelah itu sistem akan menampilkan halaman AddInterchangeActivity. Pada kelas AddInterchangeActivity, sistem akan menyimpan data *Interchange* yang dibuat oleh aktor dengan memanggil *method* submitInterchange() dan menyimpannya ke dalam database dengan menggunakan kelas DatabaseHelper. Setelah mendapatkan *result* dari proses penyimpanan data menggunakan kelas DatabaseHelper, sistem akan melakukan *update view* untuk menambahkan *marker Interchange* baru ke dalam tampilan peta.

5.1.8.15 *Sequence Diagram* Mengubah *Interchange*

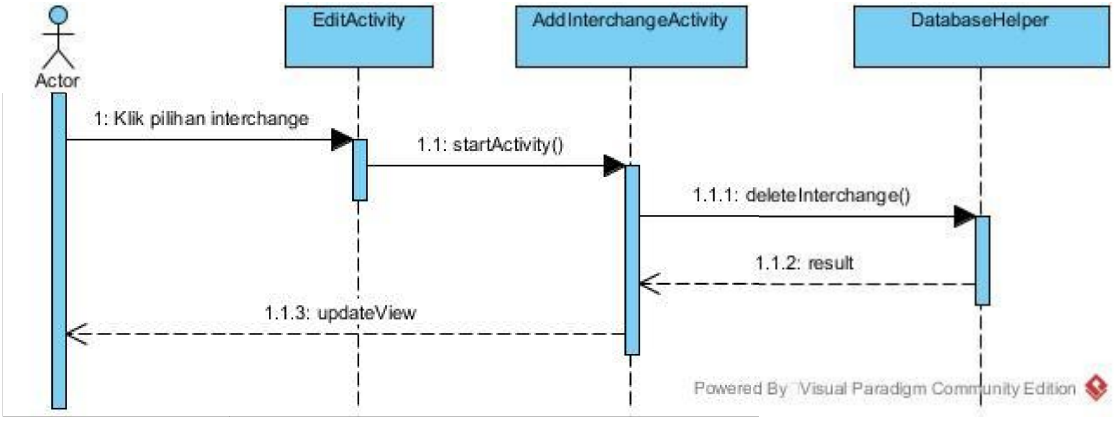


Gambar 5.30 Sequence Diagram Mengedit Interchange *Increment* 2

Pada Gambar 5.30, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Mengubah Interchange.

Pada *Sequence* Diagram Mengubah Interchange, aktor memilih terlebih dahulu pilihan mengubah *interchange* melalui EditActivity. Setelah itu sistem akan menampilkan halaman AddInterchangeActivity. Pada kelas AddInterchangeActivity, sistem akan menyimpan perubahan data *Interchange* yang dilakukan oleh aktor dengan memanggil *method* updateInterchange() dan menyimpannya ke dalam database dengan menggunakan kelas DatabaseHelper. Setelah mendapatkan *result* dari proses penyimpanan data menggunakan kelas DatabaseHelper, sistem akan melakukan *update view* untuk menyesuaikan *marker Interchange* pada tampilan peta.

5.1.8.16 *Sequence Diagram* Menghapus *Interchange*

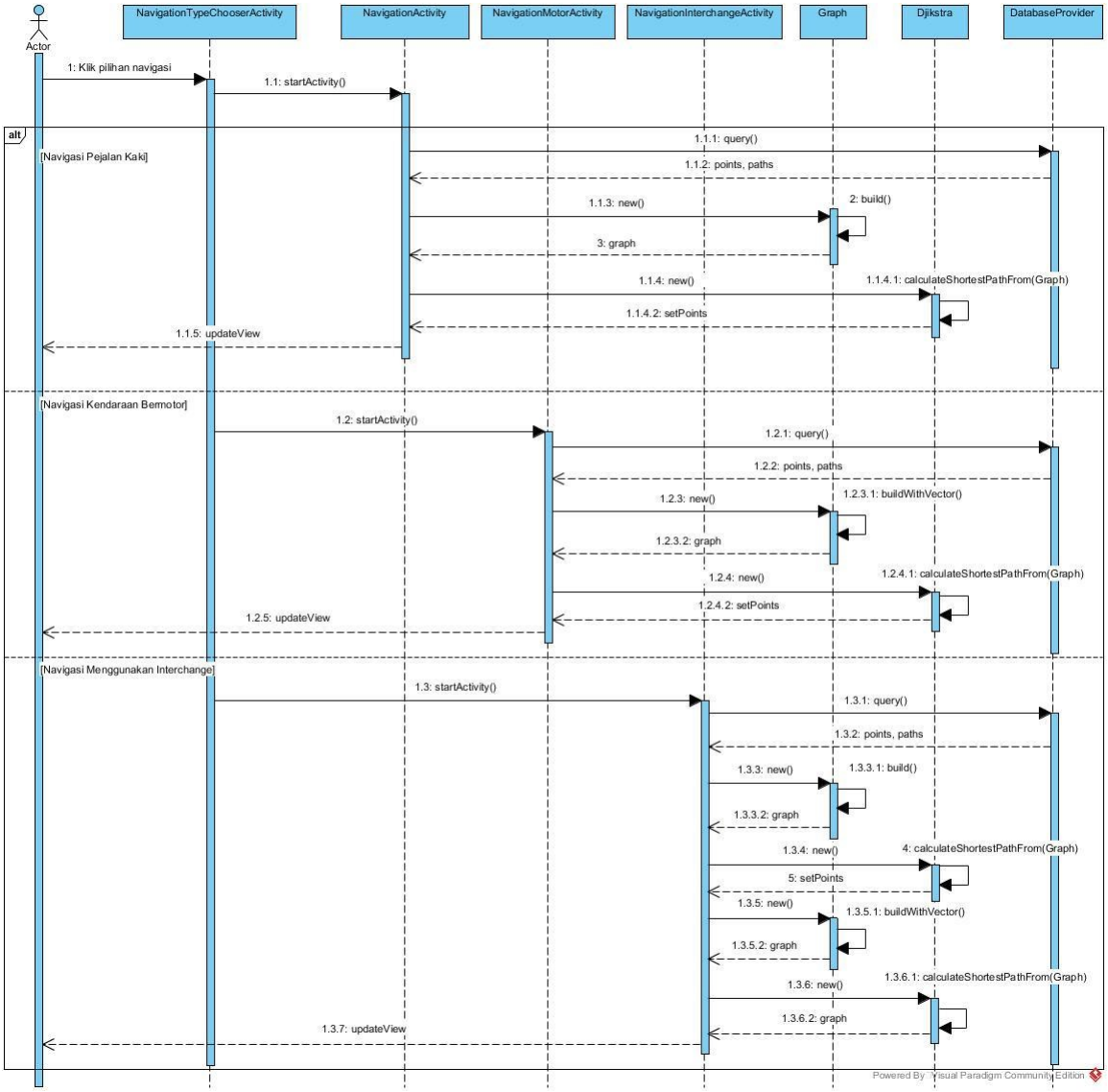


Gambar 5.31 Sequence Diagram Menghapus Interchange *Increment* 2

Pada Gambar 5.31, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Menghapus Interchange.

Pada *Sequence* Diagram Menghapus Interchange, Aktor memilih terlebih dahulu pilihan *edit interchange* melalui EditActivity. Setelah itu sistem akan menampilkan halaman AddInterchangeActivity. Pada kelas AddInterchangeActivity, sistem akan menyimpan perubahan data *Interchange* yang dilakukan oleh aktor dengan memanggil *method* deleteInterchange() dan menyimpannya ke dalam database dengan menggunakan kelas DatabaseHelper. Setelah mendapatkan *result* dari proses penyimpanan data menggunakan kelas DatabaseHelper, sistem akan melakukan *update view* untuk menghapus *marker Interchange* yang dihapus pada tampilan peta.

5.1.8.17 Sequence Diagram Melihat Rute Terdekat Ke Lokasi Tujuan Di Lingkungan Universitas Brawijaya



Gambar 5.32 Sequence Diagram Melihat Rute Terdekat Ke Lokasi Tujuan Di Lingkungan Universitas Brawijaya *Increment* 2

Pada Gambar 5.32, terdapat interaksi antara entitas yang berelasi untuk menggambarkan skenario yang terjadi dalam menjalankan fungsi pada *use case* Melihat Rute Terdekat Ke Lokasi Tujuan Di Lingkungan Universitas Brawijaya.

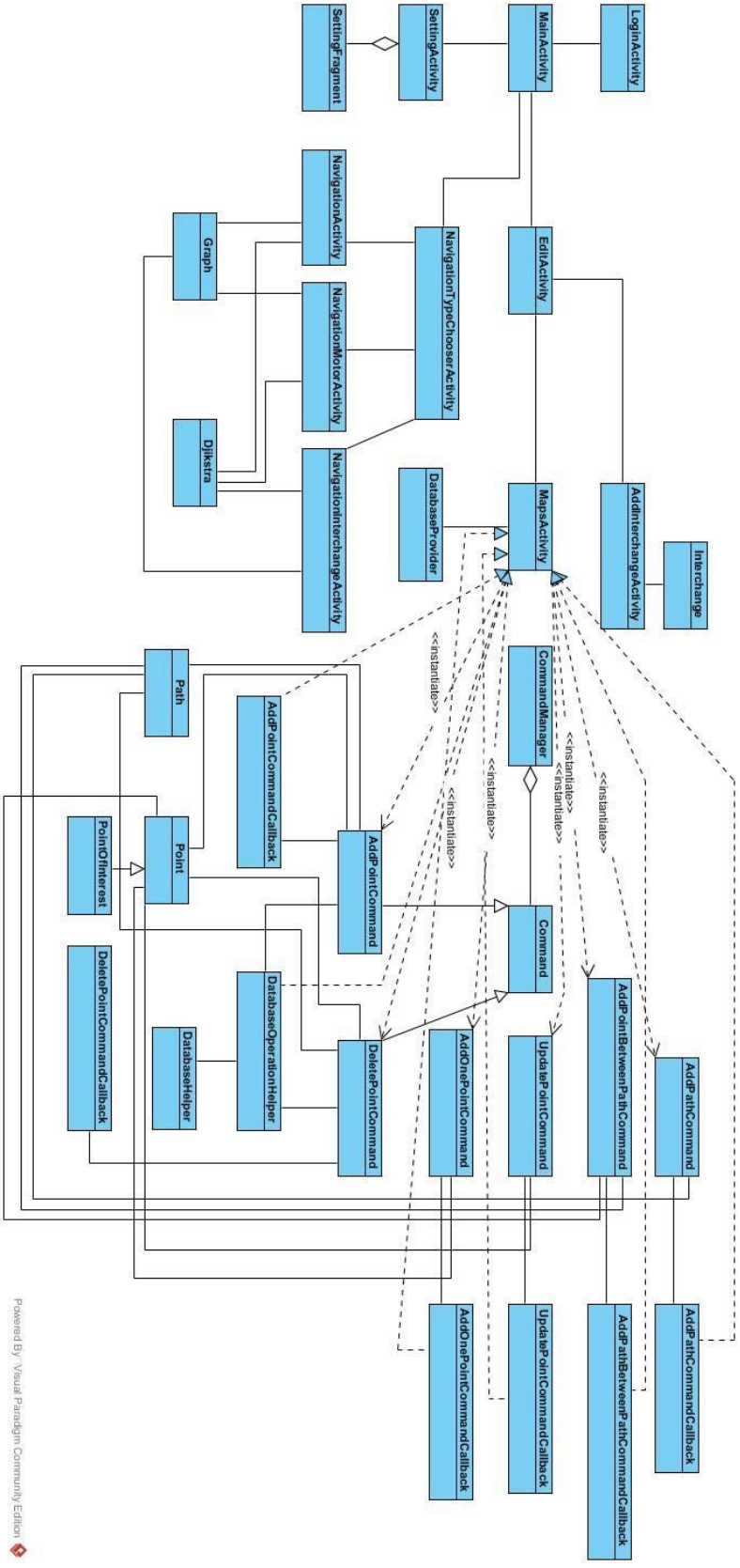
Pada *Sequence* Diagram Melihat Rute Terdekat Ke Lokasi Tujuan Di Lingkungan Universitas Brawijaya, aktor memilih salah satu jenis navigasi dari 3 jenis navigasi yang diinginkan, yaitu navigasi pejalan kaki yang terdapat pada kelas NavigationActivity, navigasi kendaraan bermotor yang terdapat pada kelas NavigationMotorActivity, dan navigasi pejalan kaki dan kendaraan bermotor menggunakan *Interchange* yang tedapat pada kelas NavigationInterchangeActivity.

Jika aktor memilih navigasi pejalan kaki, maka sistem akan mengambil data points dan paths pejalan kaki melalui Database Provider. Setelah itu, data points dan paths dibuat menjadi sebuah graf dengan memanggil method build() pad akelas Graph. Kemudian dimulai perhitungan jarak terdekat dari setiap titik pada graf ke titik lokasi tujuan dengan membuat instans kelas Djikstra baru dan memanggil method calculateShortestPath() di dalam nya dengan memasukkan Graph yang telah dibuat sebagai parameternya. Hasil dari perhitungan jarak terdekat yang dikembalikan oleh kelas Djikstra akan diolah oleh NavigationActivity untuk memberbaharui tampilan rute berdasarkan data jarak terdekat.

Jika aktor memilih navigasi kendaraan bermotor, maka sistem akan mengambil data points dan paths kendaraan bermotor melalui Database Provider. Setelah itu, data points dan paths dibuat menjadi sebuah graf dengan memanggil method build() pad akelas Graph. Kemudian dimulai perhitungan jarak terdekat dari setiap titik pada graf ke titik lokasi tujuan dengan membuat instans kelas Djikstra baru dan memanggil method calculateShortestPath() di dalam nya dengan memasukkan Graph yang telah dibuat sebagai parameternya. Hasil dari perhitungan jarak terdekat yang dikembalikan oleh kelas Djikstra akan diolah oleh NavigationMotorActivity untuk memberbaharui tampilan rute berdasarkan data jarak terdekat.

Jika aktor memilih navigasi pejalan kaki dan kendaraan bermotor menggunakan *Interchange*, maka sistem akan mengambil data points dan paths kendaraan bermotor dan pejalan kaki melalui Database Provider. Setelah itu, data points dan paths kendaraan bermotor dan pejalan kaki dibuat menjadi 2 buah graf dengan memanggil method build() pada masing-masing instans Graph. Kemudian dilakukan perhitungan jarak terdekat dari setiap titik pada graf kendaraan bermotor ke lokasi *Interchange* yang terdekat dengan lokasi tujuan. Setelah itu dilakukan juga perhitungan jarak terdekat dari setiap titik pada graf pejalan kaki ke dari lokasi *Interchange* ke lokasi tujuan. Hasil dari perhitungan jarak terdekat yang dikembalikan oleh kelas Djikstra akan diolah oleh NavigationInterchangeActivity untuk memberbaharui tampilan rute berdasarkan data jarak terdekat.

5.1.9 Perancangan *Class* Diagram *Increment* 2



Gambar 5.33 Kelas Diagram *Increment* 2

Pada rancangan kelas diagram *Increment* 2 yang digambarkan pada Gambar 5.33, terdapat 10 kelas Activity dan 1 kelas Fragment yang berinteraksi dengan pengguna. Penjelasan fungsi kelas-kelas tersebut dapat dilihat pada Tabel 5.9

Tabel 5.9 Penjelasan Fungsi Kelas Yang Berinteraksi Dengan Pengguna *Increment* 2

Nama Kelas	Fungsi
LoginActivity	Kelas yang berinteraksi dengan pengguna untuk menajalankan fungsi Login.
MainActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan halaman utama dan pilihan menu yang dapat di akses oleh pengguna.
EditActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan pilihan graf yang akan di ubah.
MapsActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan tampilan peta beserta graf yang telah dipilih oleh pengguna dan pilihan untuk menambah titik, menghapus titik, dan reset graf.
AddInterchangeActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan *interchange* yang terdapat pada *database* dan pilihan untuk menambah *interchange*, mengubah *interchange*, dan menghapus*interchange*.
NavigationTypeChooserActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan pilihan data yang akan digunakan untuk menampilkan rute terdekat ke lokasi tujuan di wilayah Universitas Brawijaya.
NavigationActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan rute terdekat ke lokasi tujuan di wilayah Universitas Brawijaya dengan menggunakan graf pejalan kaki pada *database*.
NavigationMotorActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan rute terdekat ke lokasi tujuan di wilayah Universitas Brawijaya dengan menggunakan graf kendaraan bermotor pada *database*.

NavigationInterchangeActivity	Kelas yang berinteraksi dengan pengguna untuk menampilkan rute terdekat ke lokasi tujuan di wilayah Universitas Brawijaya dengan menggunakan graf pejalan kaki, graf kendaraan bermotor, dan *interchange* pada *database*.
SettingActivity	Kelas yang menampung halaman untuk setting.
SettingFragment	Kelas yang berinteraksi dengan pengguna untuk menampilkan halaman *setting*.

Kemudian terdapat kelas yang dibuat dengan mengikuti pola perancangan *Command Pattern* yang berperan sebagai *client, receiver, invoker, command,* dan *concrete command*. Implementasi pola perancangan *Command Pattern* pada perancangan kelas diagram *increment* 2 dapat dilihat pada Tabel 5.10.

Tabel 5.10 Penjelasan Fungsi Kelas Yang Mengikut Pola Perancangan *Command Pattern Increment* 2

Peran	Implementasi
Client	Kelas yang menjadi *client* dalam pola perancangan Command Pattern adalah kelas MapsActivity. Berfungsi untuk melakukan permintaan perubahan menggunakan *Concrete Command*.
Receiver	Kelas yang menjadi *receiver* dalam pola perancangan *Command Pattern* adalah kelas DatabaseOperationHelper. Berfungsi untuk melakukan perubahan *database*.
Invoker	Kelas yang menjadi *invoker* dalam pola perancangan *Command Pattern* adalah kelas *CommandManager*. Berfungsi untuk menjalankan perintah yang diminta oleh *client*.
Command	Kelas yang menjadi *command* dalam pola perancangan *Command Pattern* adalah kelas *Command*. Berfungsi sebagai *interface* untuk *Concrete Command*.
Concrete Command	Kelas yang menjadi *concrete command* pada pola perancangan *Command Pattern* adalah kelas AddPointCommand, DeletePointCommand, AddOnePointCommand, AddPathCommand, AddPointBetweenPathCommand, dan

	UpdatePointCommand. Kelas iierfungsi untuk mendefinisikan perintah yang tersedia. Setiap *concrete command* memiliki *callback* masing-masing yang digunakan untuk melakukan perubahan tampilan ketika operasi *database* berhasil dilakukan.

Kelas lain yang terdapat pada perancangan kelas diagram *increment* 1 adalah kelas Point, Path, dan PointOfInterest yang berfungsi sebagai Model untuk data titik, jalur, dan titik tujuan. Kemudian juga terdapat kelas DatabaseHelper untuk melakukan perubahan data pada *database* dan DatabaseProvider untuk menyediakan data yang diminta melalui *query* yang dimasukkan.

5.1.10 Perancangan Algoritma *Increment* 2

Perancangan algoritma merupakan perancangan yang menjelaskan algoritma untuk menjalankan fungsionalitas sistem. Algoritma ini yang akan menjadi dasar dalam pembuatan kode pada tahap implementasi. Terdapat 3 algoritma yang dijabarkan pada perancangan algoritma *increment* 2, pemilihan 3 algoritma ini berdasarkan fungsi yang sering dilakukan dalam pemakaian aplikasi.

5.1.10.1 Perancangan Algoritma Membuat Graf

Algoritma Membuat Graf digunakan untuk membangun graf dengan menggunakan data Points dan Paths yang tersimpan dalam database. Algoritma ini menghasilkan objek Graf yang terdiri atas Points. Point pada graf ini memiliki informasi tentang tetangga terdekatnya sesuai dengan Paths yang ada. Algoritma Membuat Graf dijelaskan pada Tabel 5.11.

Tabel 5.11 Penjelasan Algoritma Membuat Graf

Nama Kelas	:	Graph
Nama Metode	:	buildWithVector()
Deskripsi	:	Metode ini merupakan method yang dijalankan ketika pengguna melakukan *tap* tombol navigasi ke suatu tempat. Pada aplikasi ini, metode akan menjalankan fungsi untuk membuat graf berdasarkan data Point dan Path yang ada pada *database*.
Algoritma	:	

```
Mulai
Ambil nilai Points dan Paths
Buat himpunan setPoints
Untuk setiap Point pada Points
```

```
        Untuk setiap Path pada Paths
                Jika latitude Point sama dengan latitude endpoint
                pada Path dan jika longitude Point sama dengan
                longitude endpoint pada Path
                    Tambah endpoint pada Path sebagai tetangga
                    dari Point
        Tambahkan Point ke dalam himpunan setPoints
Kembalikan nilai graf
Selesai
```

5.1.10.2 Perancangan Algoritma Djikstra

Algoritma *Djikstra* digunakan untuk menentukan rute terdekat berdasarkan graf yang telah dibangun. Algoritma ini menghasilkan objek Graf yang terdiri atas Points. Point pada graf ini memiliki informasi tentang rute berisi Point-Point terdekat yang harus dilalui untuk menuju ke lokasi tujuan. Algoritma Membuat Graf dijelaskan pada Tabel 5.12.

Tabel 5.12 Penjelasan Algoritma Djikstra

Nama Kelas	:	Djikstra
Nama Metode	:	calculateShortestPathFrom()
Deskripsi	:	Method ini merupakan method yang dijalankan ketika pengguna meminta rute terdekat ke lokasi tujuan di Universitas Brawijaya. Method ini digunakan untuk menjalankan fungsionalitas mencari rute terdekat dari lokasi awal hingga lokasi tujuan dengan menggunakan graf yang tersimpan pada *database*.
Algoritma	:	

```
Mulai
Ambil objek Graf
Ambil objek Point tujuan
Buat himpunan untuk menampung settledPoints
Buat himpunan untuk menampung unsettledPoint
Selama himpunan unsettledPoint kosong {
    Cari point terdekat pada himpunan unsettledPoint dan
    masukkan sebagai Current Point
    Untuk semua adjacent point dari Current Point {
        Jika himpunan settledPoints tidak mengandung adjacent
        point {
                Hitung jarak minimal dari adjacent point ke
                Current Point
```

```
                    Masukkan adjacent point ke himpunan
                    unsettledPoint
                }
    }
    Tambahkan Current Point ke dalam himpunan settledPoint
}
Atur points pada Graf dengan nilai himpunan settledPoint
Selesai
```

5.1.10.3 Perancangan Algoritma Menampilkan Rute Menggunakan 1 Jenis Graf (Graf pejalan kaki atau Graf kendaraan bermotor)

Algoritma Menampilkan Rute Menggunakan 1 Jenis Graf (Graf pejalan kaki atau Graf kendaraan bermotor) digunakan untuk menentukan rute terdekat dengan menggunakan algoritma Djikstra. Algoritma ini menghasilkan tampilan rute menuju ke lokasi tujuan. Algoritma Algoritma Menampilkan Rute Menggunakan 1 Jenis Graf (Graf pejalan kaki atau Graf kendaraan bermotor) dijelaskan pada Tabel 5.13.

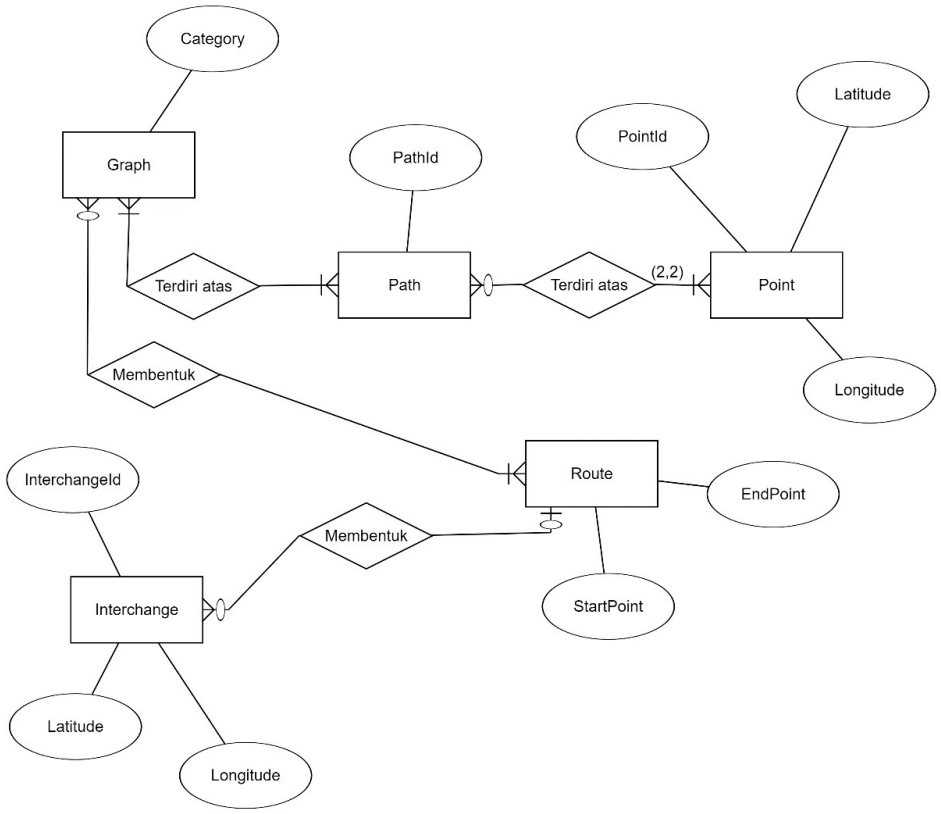
Tabel 5.13 Penjelasan Algoritma Menampilkan Rute Menggunakan 1 Jenis Graf (Graf pejalan kaki atau Graf kendaraan bermotor)

Nama Kelas	:	NavigationActivity dan NavigationMotorActivity
Nama Metode	:	generatePathFromStartTo()
Deskripsi	:	Method ini merupakan method yang dijalankan ketika pengguna meminta rute terdekat ke lokasi tujuan di Universitas Brawijaya. Method ini digunakan untuk menjalankan fungsionalitas mencari rute terdekat dari lokasi awal hingga lokasi tujuan dengan menggunakan graf yang tersimpan pada *database*. Metode ini terdapat pada kelas NavigationActivity dan NavigationMotorActivity.
Algoritma	:	

```
Mulai
Buat instans kelas Djikstra
Buat objek Point mulai
Buat objek Point tujuan akhir
Buat Graf berdasarkan titik dan jalur dari jenis graf yang
dipilih
Cari Point pada Graf yang memiliki jarak terdekat dengan Point
mulai dan masukkan sebagai Point Mulai Terdekat
Cari Point pada Graf yang memiliki jarak terdekat dengan Point
tujuan akhir dan masukkan sebagai Point Akhir Terdekat
```

```
        IF(kateogori graf yang dipilih = graf pejalan kaki)
        Membuat graf Djikstra dengan memasukkan titik tetangga
        sebagai adjacentPoint
        ELSE
        Membuat graf Djikstra dengan memasukkan titik endpoint
        dari jalur sebagai adjacentPoint
        ENDIF
Hitung rute terdekat menggunakan instans kelas Djikstra dengan
memasukkan data Graf, Point Mulai Terdekat, dan Point Akhir
Terdekat
Tampilkan rute terdekat berdasarkan graf Djikstra
Selesai
```

5.1.11 Perancangan Pemodelan Relasional Data Model *Increment* 2



Gambar 5.34 Pemodelan Relasional Data Model *Increment* 2

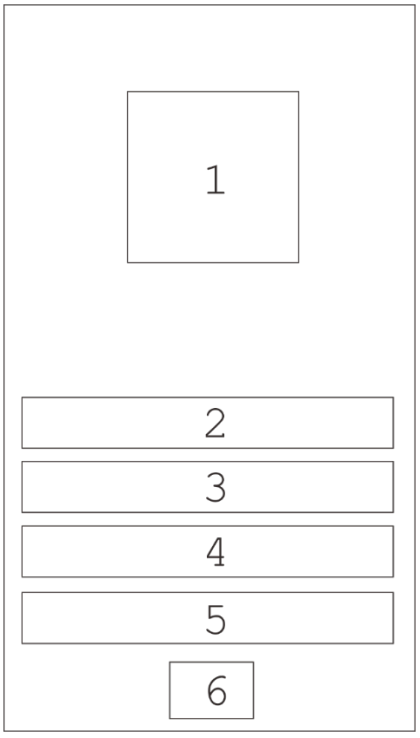
Terdapat 5 Entitas yang ditemukan pada Pemodelan Relasional Data Model *Increment* 1, yaitu Graph, Path, dan Point, Route dan Interchange. Hubungan relasi yang terjadi di antara Graph dan Path, dimana sebuah Graph terdiri atas satu atau banyak Path (*One to (One or Many)*). Path dan Point, dimana sebuah Path terdiri atas satu atau banyak Point dengan nilai minimum 2 dan maksimal 2 (*One to (One or Many)*). Graph dan Route, dimana sebuah Graph dapat membentuk satu atau banyak Route (*One to (One or Many)*). Interchange dan

Route, dimana sebuah Interchange dapat membentuk nol atau satu Route (*One to (Zero or Many*).

5.1.12 Perancangan Antarmuka *Increment* 2

5.1.12.1 Perancangan Antarmuka Halaman Login

Halaman login diakses oleh pengguna untuk memasukkan data email dan password sebagai syarat autentikasi. Halaman ini juga akan menampilkan pesan *error* jika terdapat kesalahan dalam proses autentikasi. Tidak ada perubahan pada Halaman Login pada *Increment 2* dibandingkan dengan Halaman Login pada *Increment 1*.



Gambar 5.35 Rancangan Antarmuka Halaman Login *Increment* 2

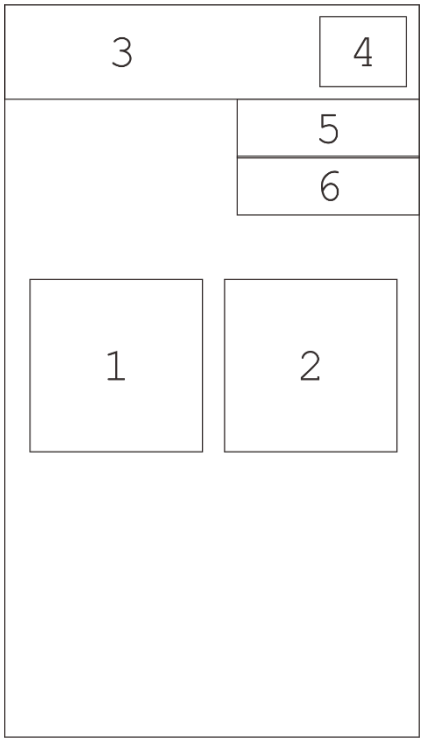
Tabel 5.14 Penjelasan Rancangan Antarmuka Halaman Login *Increment* 2

No	Nama Objek	Tipe	Keterangan
1	Application Logo	ImageView	ImageView untuk menampilkan logo aplikasi
2	Email Form	EditText	Untuk memasukkan alamat email milik pengguna
3	Password Form	EditText	Untuk memasukkan

			password akun milik pengguna
4	Submit Button	Button	Untuk melakukan proses *login*
5	Error Message	TextView	Untuk menampilkan pesan kesalahan yang terjadi ketika proses *login* dilakukan
6	Loading ProgressBar	ProgressBar	Untuk menampilkan ProgressBar ketika proses *login* sedang dilakukan

5.1.12.2 Perancangan Antarmuka Halaman Utama

Halaman utama diakses oleh pengguna ketika pengguna telah berhasil *login* ke dalam aplikasi. Pada *Increment* 2, Halaman Utama saat ini berisi pilihan untuk mengedit graf dan pilihan untuk melakukan navigasi menggunakan graf yang sudah dibuat. Selain itu, juga terdapat pilihan untuk *logout* dan untuk mengakses menu *setting* pada bagian menu.



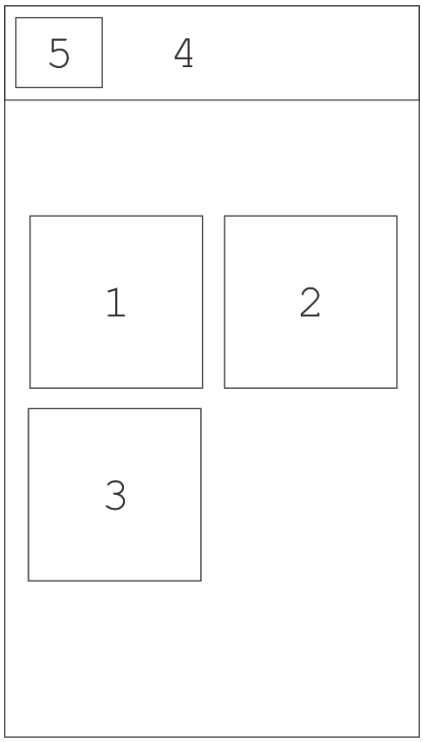
Gambar 5.36 Rancangan Antarmuka Halaman Utama *Increment* 2

Tabel 5.15 Penjelasan Rancangan Antarmuka Halaman Utama *Increment* 2

No	Nama Objek	Tipe	Keterangan
1	Opsi Edit Graf	ImageView	Untuk mengakses halaman pilih tipe graf untuk edit
2	Opsi Navigasi Menggunakan Graf	ImageView	Untuk mengakses halaman pilih tipe graf untuk navigasi
3	Action Bar	ActionBar	Untuk menampilkan nama aplikasi
4	Menu Button	MenuButton	Untuk mengakses menu item yang tersedia
5	Setting Item	MenuItem	Untuk mengakses halaman *setting*
6	Logout Item	MenuItem	Untuk keluar dari akun yang sedang *login*

5.1.12.3 Perancangan Antarmuka Halaman Pilih Tipe Graf Yang Akan Diubah

Pengguna mengakses halaman ini ketika memilih pilihan untuk mengedit graf pada Halaman Utama. Pada halaman ini, terdapat pilihan lebih lanjut untuk data yang akan di *edit*, yaitu graf pejalan kaki, graf kendaraan bermotor, dan lokasi *interchange*.



Gambar 5.37 Rancangan Antarmuka Halaman Pilih Tipe Graf Yang Akan Diubah *Increment* 2

Tabel 5.16 Penjelasan Rancangan Antarmuka Halaman Pilih Tipe Graf Yang Akan Diubah *Increment* 2

No	Nama Objek	Tipe	Keterangan
1	Opsi Graf Pejalan Kaki	ImageView	Untuk membuka halaman untuk mengedit graf pejalan kaki
2	Opsi Graf Kendaraan Bermotor	ImageView	Untuk membuka halaman untuk mengedit graf kendaraan bermotor

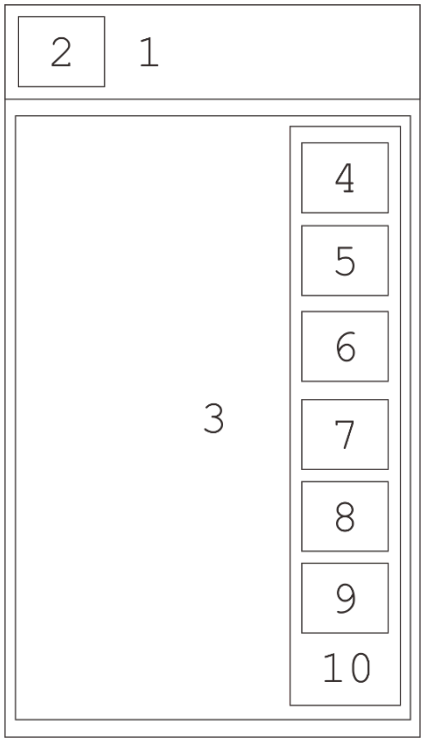
3	Opsi Lokasi Interchange	ImageView	Untuk membuka halaman untuk mengedit lokasi *interchange*
4	Action Bar	ActionBar	Untuk menampilkan konteks pilihan opsi
5	Back Button	MenuButton	Untuk kembali ke halaman utama

5.1.12.4 Perancangan Antarmuka Halaman Mengubah Graf

Halaman edit graf di *reuse* untuk menampilkan halaman edit graf untuk semua tipe graf. Oleh karena itu, tipe graf pejalan kaki dan tipe graf kendaraan bermotor menggunakan *resource* yang sama untuk menampilkan halaman edit graf. Pada halaman edit graf, terdapat Fragment peta yang memunculkan peta Google Maps. Di dalam Fragment peta ini terdapat Marker dan Polyline yang membentuk graf.

Pengguna dapat melakukan *tap* di area Fragment peta untuk memasukkan titik baru. Pengguna juga dapat melakukan *pinch* pada tampilan peta untuk melakukan perbesaran dan pengecilan tampilan peta. Selain itu, pengguna juga dapat melakukan *tap* pada Marker untuk memilih Marker tersebut.

Di sisi sebelah kanan terdapat panel tombol sebagai pendukung bagi pengguna ketika sedang melakukan proses mengedit Graf. Panel tombol berisi tombol-tombol pendukung yaitu tombol *undo*¸ tombol *redo*, tombol hapus, dan tombol *reset*. Pada *Increment* 2, terdapat tambahan tombol pada panel tombol untuk menyesuaikan kebutuhan yang ada pada pengembangan aplikasi *Increment* 2. Penambahan tombol yang dimaksud yaitu tombol untuk menyatukan jalur dan tombol untuk membuat titik baru diluar graf yang telah dibuat.



Gambar 5.38 Rancangan Antarmuka Halaman Mengubah Graf *Increment* 2

Tabel 5.17 Penjelasan Rancangan Antarmuka Halaman Mengubah Graf *Increment* 2

No	Nama Objek	Tipe	Keterangan
1	Action Bar	ActionBar	Untuk menampilkan nama graf yang sedang di edit
2	Back Button	MenuButton	Untuk kembali ke halaman utama
3	Map Fragment	Fragment	Untuk menampilkan tampilan peta Google Maps
4	Undo Button	FloatingActionButton	Untuk melakukan aksi *undo*
5	Redo Button	FloatingActionButton	Untuk melakukan aksi *redo*
6	Join Button	FloatingActionButton	Untuk menyatukan 2 titik

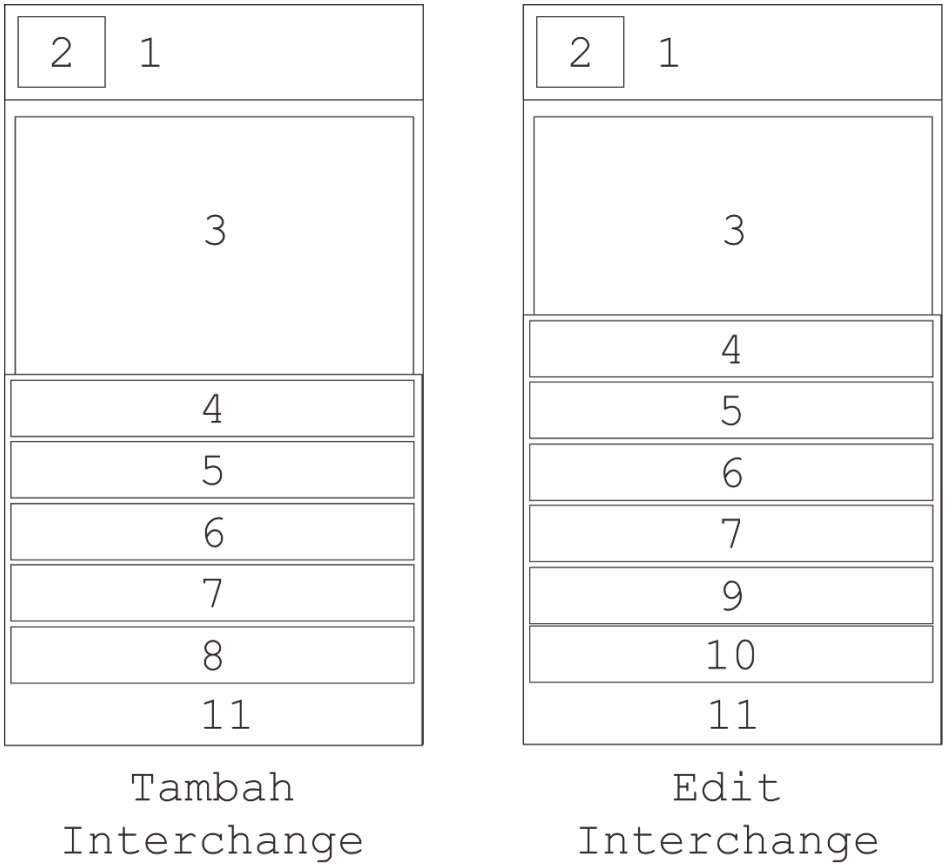
7	Add Button	FloatingActionButton	Untuk menambahkan titik baru diluar graf yang telah dibuat
8	Delete Button	FloatingActionButton	Untuk menghapus suatu titik pada graf
9	Reset Button	FloatingActionButton	Untuk melakukan aksi *reset*, yaitu menghapus semua titik dan jalur yang terdapat suatu tipe graf
10	Panel Button	LinearLayout	Sebagai *panel* atau tempat untuk menampung tombol-tombol pendukung dalam melakukan *edit* graf

5.1.12.5 Perancangan Antarmuka Halaman Mengubah Interchange

Pengguna mengakses Halaman Edit Interchange untuk melakukan penamabahan titik *interchange* dan melakukan perubahan serta penghapusan terhadap *interchange* yang sudah ada. Pada halaman ini, pengguna akan melihat sebuah Fragment peta Google Maps yang didalamnya terdapat Marker sebagai bentuk *interchange* yang ada pada *database*.

Ketika pengguna melakukan *tap* pada area peta, maka sistem akan melakukan *expand* terhadap BottomSheet yang berisi form untuk memasukkan *interchange* baru.

Ketika pengguna melakukan *tap* pada Marker yang ada di tampilan peta, maka sistem akan melakukan *expand* terhadap BottomSheet yang berisi form yang telah terisi data *interchange*. Pada form ini, pengguna dapat mengedit atau melakukan penghapusan terhadap data yang dipilih.



Gambar 5.39 Rancangan Antarmuka Halaman Mengubah Interchange *Increment* 2

Tabel 5.18 Penjelasan Rancangan Antarmuka Halaman Mengubah Interchange *Increment* 2

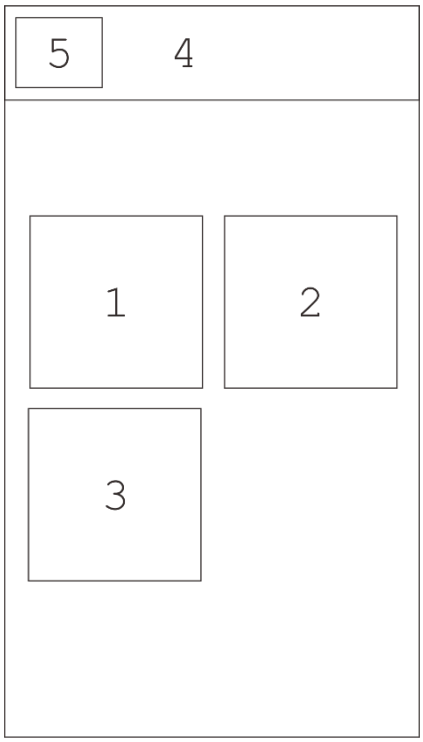
No	Nama Objek	Tipe	Keterangan
1	Action Bar	ActionBar	Untuk menampilkan jenis navigasi yang sedang diproses
2	Back Button	MenuButton	Untuk kembali ke menu pilihan tipe graf untuk navigasi
3	Map Fragment	Fragment	Untuk menampilkan tampilan peta Google Maps

4	Interchange Name Form	EditText	Untuk menampilkan dan sebagai *input* nama *interchange*
5	Interchange Description Form	EditText	Untuk menampilkan dan sebagai *input* deskripsi *interchange*
6	Interchange Type Form	Spinner	Untuk menampilkan dan sebagai *input* tipe *interchange*
7	Interchange Availability Form	Switch	Untuk menampilkan dan sebagai *input* status ketersediaan *interchange*
8	Add Interchange Button	Button	Untuk menambahkan *interchange* baru ke *database*
9	Update Interchange Button	Button	Untuk memperbaharui *interchange* pada *database*
10	Delete Interchange Button	Button	Untuk menghapus *interchange* dari *database*
11	Bottom Sheet	BottomSheet	Untuk menampung *widget* pendukung

5.1.12.6 Perancangan Antarmuka Halaman Pilih Tipe Graf Untuk Navigasi

Pengguna mengakses halaman ini ketika memilih pilihan untuk melakukan navigasi menggunakan graf pada Halaman Utama. Pada halaman ini, terdapat pilihan lebih lanjut untuk data yang akan digunakan untuk navigasi, yaitu graf

pejalan kaki, graf kendaraan bermotor, dan gabungan dari graf pejalan kaki, graf kendaraan bermotor, *interchange*.



Gambar 5.40 Rancangan Antarmuka Halaman Pilih Tipe Graf Untuk Navigasi *Increment* 2

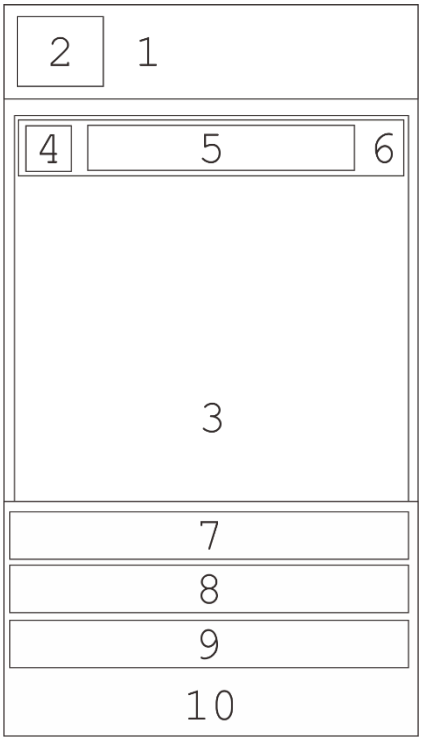
Tabel 5.19 Penjelasan Rancangan Antarmuka Halaman Pilih Tipe Graf Untuk Navigasi *Increment* 2

No	Nama Objek	Tipe	Keterangan
1	Opsi Graf Pejalan Kaki	ImageView	Untuk membuka halaman untuk navigasi menggunakan graf pejalan kaki
2	Opsi Graf Kendaraan Bermotor	ImageView	Untuk membuka halaman untuk navigasi menggunakan graf kendaraan bermotor
3	Opsi Gabungan Graf Pejalan Kaki, Graf Kendaraan Bermotor, dan *interchange*	ImageView	Untuk membuka halaman untuk navigasi menggunakan

			gabungan graf pejalan kaki, graf kendaraan bermotor, dan *interchange*
4	Action Bar	ActionBar	Untuk menampilkan konteks pilihan opsi
5	Back Button	MenuButton	Untuk kembali ke halaman utama

5.1.12.7 Perancangan Antarmuka Halaman Navigasi

Pengguna mengakses halaman ini ketika memilih pilihan untuk melakukan navigasi menggunakan graf pada Halaman Utama, dan telah memilih data yang akan digunakan untuk navigasi pada Halaman Pilih Tipe Graf Untuk Navigasi. Pada halaman ini, terdapat Fragment yang berfungsi untuk menampilkan tampilan peta Google Maps. Pada Fragment tersebut, terdapat Marker yang menandai lokasi-lokasi yang ada di wilayah Universitas Brawijaya. Marker-marker yang ada pada tampilan peta dapat di *tap* oleh user untuk memunculkan BottomSheet yang berisi informasi nama dan deskripsi lokasi. Selain itu, pada BottomSheet juga terdapat tombol untuk memulai perhitungan rute terdekat menggunakan algoritma Djikstra dan data yang telah dipilih oleh pengguna. Perhitungan rute terdekat juga dapat dilakukan dengan menggunakan SearchView yang ada pada posisi atas tampilan peta, yaitu dengan memasukkan nama lokasi yang ingin dituju dan melakukan *tap* pada hasil pencarian lokasi.



Gambar 5.41 Rancangan Antarmuka Halaman Navigasi *Increment* 2

Tabel 5.20 Penjelasan Rancangan Antarmuka Halaman Navigasi *Increment* 2

No	Nama Objek	Tipe	Keterangan
1	Action Bar	ActionBar	Untuk menampilkan tipe navigasi yang dipilih
2	Back Button	MenuButton	Untuk kembali ke halaman pilihan data untuk navigasi
3	Map Fragment	Fragment	Untuk menampilkan tampilan peta Google Maps
4	Search Button	SearchButton	Untuk memunculkan virtual keyboard untuk menuliskan *query* pencarian lokasi

5	Query Edit Text	EditText	Untuk memasukkan *query* pencarian lokasi
6	Search View	SearchView	Untuk menampilkan tampilan pencarian
7	Name Text View	TextView	Untuk menampilkan nama lokasi
8	Description Text View	TextView	Untuk menampilkan deskripsi lokasi
9	Calculate Route Button	Button	Untuk memulai perhitungan jarak terdekat menggunakan algoritma djikstra dan data yang telah dipilih oleh pengguna
10	Bottom Sheet	BottomSheet	Untuk menampung *widget* pendukung

5.2 Implementasi

Pada tahapan implementasi di setiap *increment*, dilakukan implementasi terhadap perancangan yang telah dibuat pada tahapan perancangan. Implementasi yang dilakukan adalah implementasi sistem, implementasi kode, implementasi basis data, dan implementasi antarmuka.

5.2.1 Spesifikasi Sistem *Increment* 1

Spesifikasi sistem yang digunakan untuk membangun sistem ini meliputi spesifikasi perangkat keras, perangkat lunak, dan sistem operasi,

Spesifikasi Perangkat Keras

Pengembangan Aplikasi Mobile Struktur Data Graf menggunakan perangkat keras berupa komputer dengan spesifikasi yang dijelaskan pada tabel berikut :

Nama Komponen	Spesifikasi
Processor	Intel Core i7-6700
Memory	8 gb DDR4
Display	NVIDIA GeForce GTX 1050 Ti
Hard Disk	Samsung 850 Evo SSD 250 gb

Spesifikasi Perangkat Lunak

Pengembangan Aplikasi Mobile Struktur Data Graf menggunakan perangkat keras berupa komputer dengan spesifikasi yang dijelaskan pada tabel berikut :

Nama Komponen	Spesifikasi
***Editor* Perancangan**	Visual Paradigm, ERDPlus, CorelDraw X8
***Editor* Pemrograman**	Android Studio
Framework	Android
Bahasa Pemrograman	Java
DBMS (*Database Management System*)	SQLite

Spesifikasi Sistem Operasi

Pengembangan Aplikasi Mobile Struktur Data Graf menggunakan perangkat keras berupa komputer sistem operasi Windows 10 Enterprise 64-bit.

5.2.2 Implementasi Kode *Increment* 1

Implementasi kode merupakan penjelasan implementasi algoritma yang didefinisikan pada tahap perancangan menjadi kode yang digunakan untuk menjalankan fungsionalitas aplikasi. Algoritma di implementasikan menjadi kode dalam bahasa Java.

5.2.2.1 Implementasi Kode Menampilkan Titik Dan Jalur Dari Graf Yang Dipilih

Pada tahap ini, dilakukan implementasi kode pada algoritma Menampilkan Titik Dan Jalur Dari Graf Yang Dipilih yang dijelaskan pada Tabel 5.3. Implementasi kode Menampilkan Titik Dan Jalur Dari Graf Yang Dipilih dijelaskan pada Tabel 5.21.

Tabel 5.21 Penjelasan Implementasi Kode Menampilkan Titik Dan Jalur Dari Graf Yang Dipilih

Nama Kelas	:	MapsActivity
Nama Metode	:	onMapReady()
Deskripsi	:	Method ini merupakan method yang dijalankan ketika tampilan peta pada halaman mengubah graf telah siap untuk menerima interaksi dari pengguna. Method ini berfungsi untuk menyiapkan penangkap interaksi pada peta dan memicu *background task* untuk mengambil data titik dan jalur pada *database*.
Kode	:	

```
   @Override
     public void onMapReady(GoogleMap googleMap) {
         mMap = googleMap;

getSupportLoaderManager().restartLoader(LOADER_POINT_ID, null,
pointsLoaderCallback);
         mMap.setOnMapClickListener(this);
         mMap.setOnMarkerClickListener(this);
     }
```

5.2.2.2 Implementasi Kode Implementasi Kode Menambah Titik Dalam Graf

Pada tahap ini, dilakukan implementasi kode pada algoritma Menambah Titik Dalam Graf yang dijelaskan pada Tabel 5.4. Implementasi kode Menambah Titik Dalam Graf dijelaskan pada Tabel 5.21.

Tabel 5.22 Penjelasan Implementasi Kode Menambah Titik Dalam Graf

Nama Kelas	:	MapsActivity
Nama Metode	:	onMapClikListener()
Deskripsi	:	Metode ini merupakan method yang dijalankan ketika pengguna melakukan *tap* pada tampilan peta. Dalam aplikasi ini, metode akan menjalankan fungsionalitas untuk mengubah data graf dengan menambah titik dalam graf pada lokasi peta yang di *tap* oleh pengguna.
Kode	:	

```
@Override
    public void onMapClick(LatLng latLng) {
            if(selectedPosition == null){
                SnackbarUtil.showSnackBar(root_map,
snackbar,"Pilih point terlebih dahulu.",
Snackbar.LENGTH_LONG);
            } else{
                commandManager.doCommand(
                        new AddPointCommand(
                                this,
                                mDbHelper,
                                selectedPosition,
                                latLng,
                                PATH_CATEGORY));

            }
        }
```

5.2.2.3 Implementasi Kode Implementasi Kode Menghapus Titik

Pada tahap ini, dilakukan implementasi kode pada algoritma Menghapus Titik yang dijelaskan pada Tabel 5.5. Implementasi kode Menghapus Titik dijelaskan pada Tabel 5.23.

Tabel 5.23 Penjelasan Implementasi Kode Menghapus Titik

Nama Kelas	:	MapsActivity
Nama Metode	:	deletePoint()

Deskripsi	:	Metode ini merupakan method yang dijalankan ketika pengguna melakukan *tap* pada tombol hapus. Dalam aplikasi ini, metode akan menjalankan fungsionalitas untuk menghapus titik yang telah dipilih oleh pengguna.
Kode	:	

```
public void deletePoint(View view) {
    commandManager.doCommand(new DeletePointCommand(
            this,
            mDbHelper,
            selectedPosition,
            paths,
            PATH_CATEGORY));
}
```

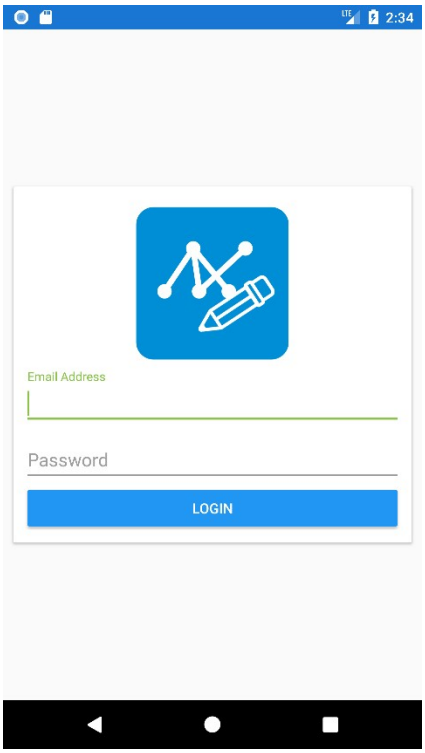
5.2.3 Implementasi Basis Data *Increment* 1

Gambar 5.42 Implementasi Basis Data *Increment* 1

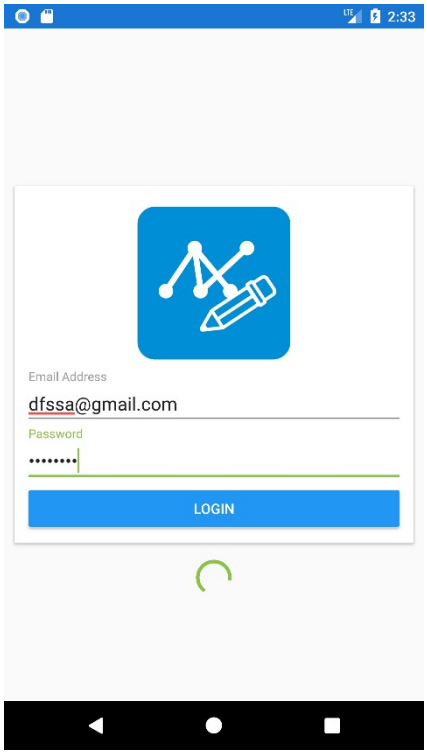
Pada implementasi data, diperoleh table-tabel dari *database* yang digunakan untuk menjalankan fungsionalitas sistem. Tabel-tabel tersebut adalah tabel Point dan tabel Path. Tabel point memiliki relasi *one to many* dengan tabel Path, karena Point dapat tergabung dalam beberapa Path.Implementasi antarmuka ini dilakukan berdasarkan perancangan antarmuka yang dibuat pada bab perancangan sebelumnya.

5.2.4 Implementasi Antarmuka *Increment* 1

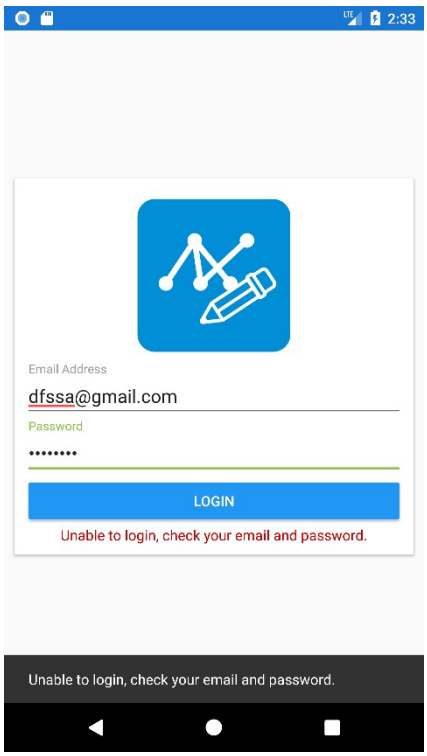
5.2.4.1 Implementasi Antarmuka Halaman Login



Gambar 5.43 Implementasi Antarmuka Halaman Login *Increment* 1



Gambar 5.44 Implementasi Antarmuka Halaman Login *Increment* 1 Status *Loading*



Gambar 5.45 Implementasi Antarmuka Halaman Login *Increment* 1 Status Gagal

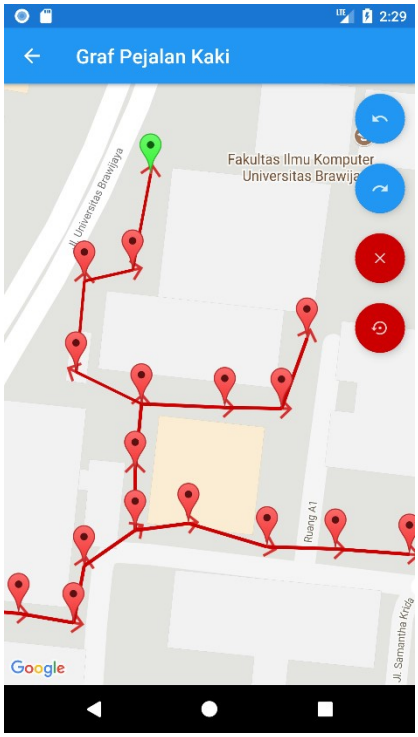
Pada implementasi antarmuka halaman login, pengguna dapat memasukkan email dan password melalui form input berupa Edit Text yang ada. Setelah itu pengguna melakukan *tap* terhadap tombol *login* untuk memulai proses *login*. Proses *login* berlangsung ditandai dengan munculnya Progress Bar. Setelah itu pengguna akan dialihkan ke halaman utama jika proses *login* berhasil, dan akan mendapatkan notifikasi jika proses *login* gagal.

5.2.4.2 Implementasi Antarmuka Halaman Utama

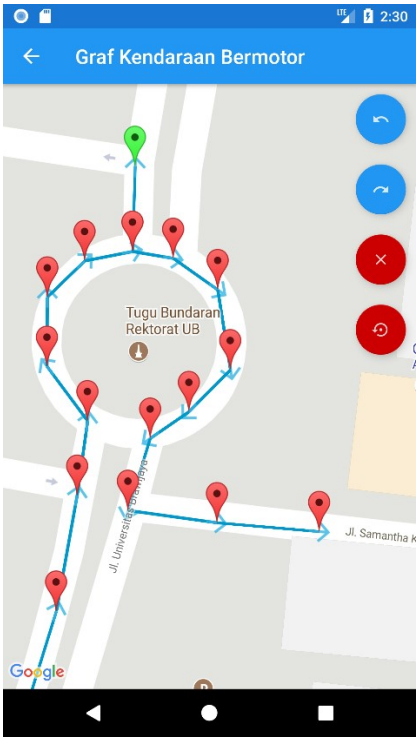
Gambar 5.46 Implementasi Antarmuka Halaman Utama *Increment* 1

Pada implementasi antarmuka halaman utama *increment* 1, terdapat pilihan untuk mengedit graf pejalan kaki yang ditampilkan dengan *icon* pejalan kaki, dan graf kendaraan bermotor yang ditampilkan dengan *icon* motor dan mobil. Selain itu, pengguna juga bisa mengakses halaman Setting dan Logout melalui Menu Button pada pojok kiri atas.

5.2.4.3 Implementasi Antarmuka Halaman Edit Graf



Gambar 5.47 Implementasi Antarmuka Halaman Edit Graf Pejalan Kaki *Increment* 1



Gambar 5.48 Implementasi Antarmuka Halaman Edit Graf Kendaraan Bermotor *Increment* 1

Pada implementasi antarmuka halaman edit graf *increment* 1, terdapat 2 kemungkinan yang terjadi, yaitu pengguna memilih graf pejalan kaki atau graf kendaraan bermotor. Tampilan halaman di dominasi oleh tampilan peta Google Maps, yang didalam nya terdapat kumpulan titik yang ditampilkan sebagai Marker dan kumpulan jalur yang ditampilkan sebagai Polyline. Titik dan jalur didapat berdasarkan pilihan graf yang dipilih oleh pengguna.

Pada graf pejalan kaki, jalur graf akan berwarna merah dan pada graf kendaraan bermotor, jalur graf akan berwarna biru, hal ini dilakukan untuk membedakan tampilan antara graf pejalan kaki dan graf kendaraan bermotor. Di sisi kanan layar terdapat panel yang berisi tombol-tombol pendukung yaitu tombol *undo*, tombol *redo*, tombol hapus, dan tombol *reset*. Tombol-tombol tersebut berfungsi untuk mendukung pengguna dalam melakukan perubahan pada graf yang diplih. Fungsi tombol-tombol tersebut adalah :

Tombol *undo* : Mengembalikan proses yang telah dilakukan sebanyak 1 langkah

Tombol *redo* : Mengembalikan proses yang dilakukan undo sebanyak 1 langkah

Tombol hapus : Menghapus suatu titik

Tombol *reset* : Menghapus seluruh titik dan jalur yang terdapat dalam suatu graf

5.2.5 Spesifikasi Sistem *Increment* 2

Spesifikasi sistem yang digunakan untuk membangun sistem ini meliputi spesifikasi perangkat keras, perangkat lunak, dan sistem operasi,

Spesifikasi Perangkat Keras

Pengembangan Aplikasi Mobile Struktur Data Graf menggunakan perangkat keras berupa komputer dengan spesifikasi yang dijelaskan pada tabel berikut :

Nama Komponen	Spesifikasi
Processor	Intel Core i7-6700
Memory	8 gb DDR4
Display	NVIDIA GeForce GTX 1050 Ti
Hard Disk	Samsung 850 Evo SSD 250 gb

Spesifikasi Perangkat Lunak

Pengembangan Aplikasi Mobile Struktur Data Graf menggunakan perangkat keras berupa komputer dengan spesifikasi yang dijelaskan pada tabel berikut :

Nama Komponen	Spesifikasi
***Editor* Perancangan**	Visual Paradigm, ERDPlus, CorelDraw X8

***Editor* Pemrograman**	Android Studio
Framework	Android
Bahasa Pemrograman	Java
DBMS (*Database Management System*)	SQLite

Spesifikasi Sistem Operasi

Pengembangan Aplikasi Mobile Struktur Data Graf menggunakan perangkat keras berupa komputer sistem operasi Windows 10 Enterprise 64-bit.

5.2.6 Implementasi Kode *Increment* 2

Implementasi kode merupakan penjelasan implementasi algoritma yang didefinisikan pada tahap perancangan menjadi kode yang digunakan untuk menjalankan fungsionalitas aplikasi. Algoritma di implementasikan menjadi kode dalam bahasa Java.

5.2.6.1 Implementasi Kode Membuat Graf

Pada tahap ini, dilakukan implementasi kode pada algoritma Membuat Graf yang dijelaskan pada Tabel 5.11. Implementasi kode Menghapus Titik dijelaskan pada Tabel 5.24.

Tabel 5.24 Penjelasan Implementasi Kode Membuat Graf

Nama Kelas	:	Graph
Nama Metode	:	buildWithVector ()
Deskripsi	:	Metode ini merupakan method yang dijalankan ketika pengguna melakukan *tap* tombol navigasi ke suatu tempat. Pada aplikasi ini, metode akan menjalankan fungsi untuk membuat graf berdasarkan data Point dan Path yang ada pada *database*.
Kode	:	

```
Set<Point> setPoints = new HashSet<>();
        for(Point p: points) {
            for(Path path: paths) {
              if(p.latitude == path.endPoint.latitude &&
p.longitude == path.endPoint.longitude)

p.addDestination(findPoint(path.startPoint, points),
Helper.calculateDistance(p, path.startPoint));
```

```
            }
            setPoints.add(p);
        }

        return setPoints;
```

5.2.6.2 Implementasi Kode Algoritma Djikstra

Pada tahap ini, dilakukan implementasi kode pada algoritma Djikstra yang dijelaskan pada Tabel 5.12. Implementasi kode Djikstra dijelaskan pada Tabel 5.25.

Tabel 5.25 Penjelasan Implementasi Kode Algoritma Djikstra

Nama Kelas	:	Djikstra
Nama Metode	:	calculateShortestPathFrom()
Deskripsi	:	Method ini merupakan method yang dijalankan ketika pengguna meminta rute terdekat ke lokasi tujuan di Universitas Brawijaya. Method ini digunakan untuk menjalankan fungsionalitas mencari rute terdekat dari lokasi awal hingga lokasi tujuan dengan menggunakan graf yang tersimpan pada *database*.
Kode	:	

```
public Graph calculateShortestPathFrom(Graph graph, Point
source) {

        source.setDistance(0D);

        Set<Point> settledPoints = new HashSet<>();
        Set<Point> unsettledPoints = new HashSet<>();
        unsettledPoints.add(source);

        while (unsettledPoints.size() != 0) {
            Point currentPoint =
getLowestDistancePoint(unsettledPoints);
            unsettledPoints.remove(currentPoint);
            for (Map.Entry<Point, Double> adjacencyPair:

currentPoint.getAdjacentPoints().entrySet()) {
                Point adjacentPoint = adjacencyPair.getKey();
                Double edgeWeight = adjacencyPair.getValue();
```

```
                if (!settledPoints.contains(adjacentPoint)) {
                    calculateMinimumDistance(adjacentPoint,
edgeWeight, currentPoint);
                    unsettledPoints.add(adjacentPoint);
                }
            }
            settledPoints.add(currentPoint);
        }
        graph.setPoints(settledPoints);
        return graph;
    }
```

5.2.6.3 Implementasi Kode Menampilkan Rute Menggunakan 1 Jenis Graf (Graf pejalan kaki atau Graf kendaraan bermotor)

Pada tahap ini, dilakukan implementasi kode pada algoritma Membuat Graf yang dijelaskan pada Tabel 5.13. Implementasi kode Implementasi Kode Menampilkan Rute Menggunakan 1 Jenis Graf (Graf pejalan kaki atau Graf kendaraan bermotor) dijelaskan pada Tabel 5.26.

Tabel 5.26 Penjelasan Implementasi Kode Menampilkan Rute Menggunakan 1 Jenis Graf (Graf pejalan kaki atau Graf kendaraan bermotor)

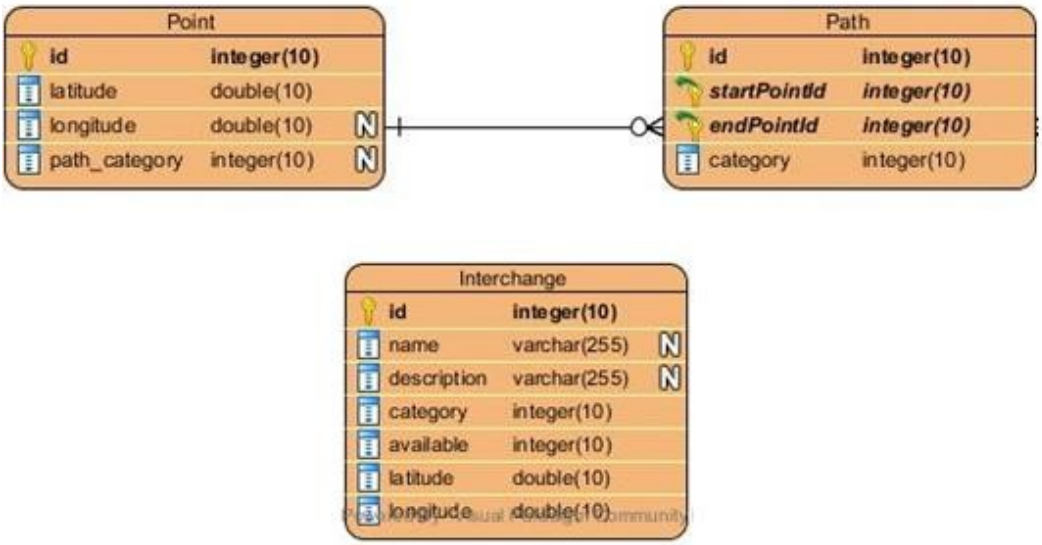
Nama Kelas	:	NavigationActivity dan NavigationMotorActivity
Nama Metode	:	generatePathFromStartTo()
Deskripsi	:	Method ini merupakan method yang dijalankan ketika pengguna meminta rute terdekat ke lokasi tujuan di Universitas Brawijaya. Method ini digunakan untuk menjalankan fungsionalitas mencari rute terdekat dari lokasi awal hingga lokasi tujuan dengan menggunakan graf yang tersimpan pada *database*. Metode ini terdapat pada kelas NavigationActivity dan NavigationMotorActivity.
Algoritma	:	

```
private void generatePathFromStartTo(LatLng latLng) {
        djikstraPoints.clear();
        djikstraPaths.clear();
        convertDjikstraResources();
        Dijkstra dijkstra = new Dijkstra();
        Graph graph = new Graph();
```

```
        devfikr.skripsi.ubnav.djikstra.base.Point pointStart =
                new devfikr.skripsi.ubnav.djikstra.base.Point(
                        "Start",
                        startLatLng.latitude,
                        startLatLng.longitude
                );
        devfikr.skripsi.ubnav.djikstra.base.Point pointDest =
                new devfikr.skripsi.ubnav.djikstra.base.Point(
                        "Destination",
                        latLng.latitude,
                        latLng.longitude
                );
        Set<devfikr.skripsi.ubnav.djikstra.base.Point>
pointsSet;


        pointsSet
                = Graph.build(djikstraPoints, djikstraPaths);


        devfikr.skripsi.ubnav.djikstra.base.Point
closestPointFromStart =
                getClosestNode(pointStart, pointsSet);
        devfikr.skripsi.ubnav.djikstra.base.Point
closestPointFromDest =
                getClosestNode(pointDest, pointsSet);
        graph.setPoints(pointsSet);
        try {


            Graph djikstraGraph;


            if(navCategory ==
DatabaseContract.PathColumns.CATEGORY_WALKING){
                djikstraGraph =
dijkstra.calculateShortestPathFrom(
                        graph,
                        closestPointFromDest
                );
            } else{
                djikstraGraph =
dijkstra.calculateShortestVectorPathFrom(
                        graph,
```

```
                        closestPointFromDest,
                        djikstraPaths
                    );
                }
                generatePath(djikstraGraph,pointStart,
closestPointFromStart, closestPointFromDest, pointDest);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

5.2.7 Implementasi Basis Data *Increment* 2



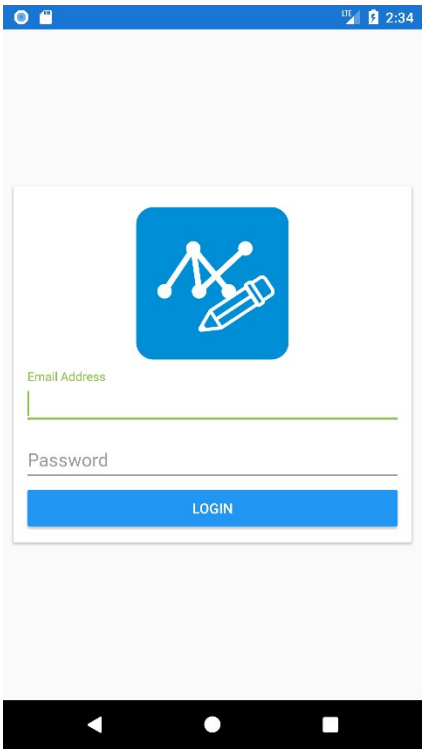
Gambar 5.49 Implementasi Basis Data *Increment* 2

Pada implementasi data, diperoleh table-tabel dari *database* yang digunakan untuk menjalankan fungsionalitas sistem. Tabel-tabel tersebut adalah tabel Point dan tabel Path. Tabel point memiliki relasi *one to many* dengan tabel Path, karena Point dapat tergabung dalam beberapa Path. Terdapat tambahan tabel Interchange pada *increment* 2 untuk menyimpan titik-titik lokasi *interchange* beserta penjelasan lokasi tersebut. Tabel *interchange* tidak memiliki relasi dengan tabel lainnya.

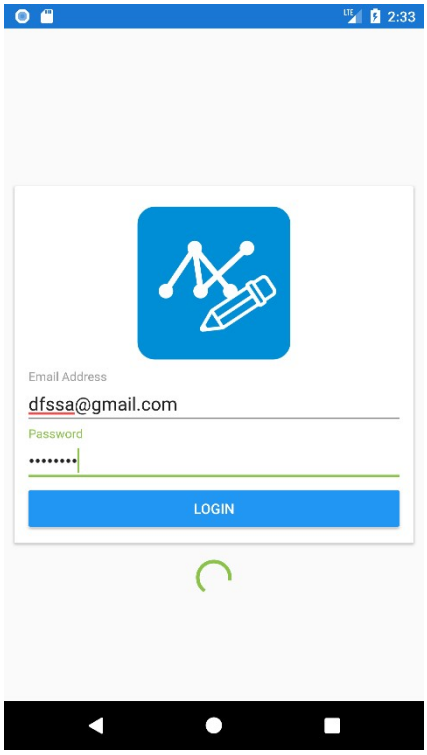
5.2.8 Implementasi Antarmuka *Increment* 2

Implementasi antarmuka ini dilakukan berdasarkan perancangan antarmuka yang dibuat pada bab perancangan sebelumnya.

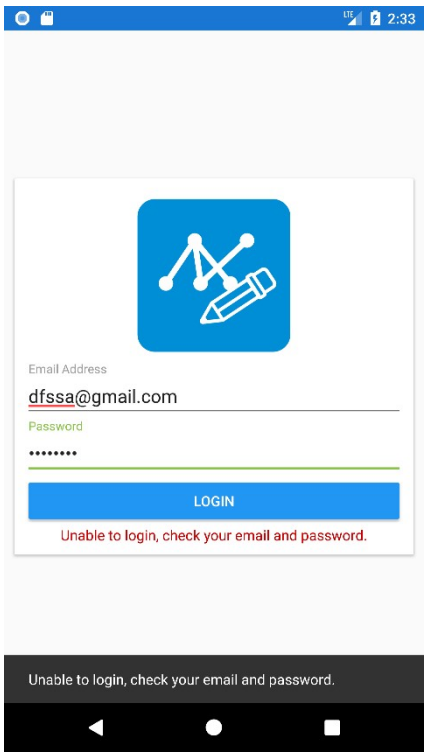
5.2.8.1 Implementasi Antarmuka Halaman Login



Gambar 5.50 Implementasi Antarmuka Halaman Login *Increment* 2



Gambar 5.51 Implementasi Antarmuka Halaman Login *Increment* 2 Status *Loading*



Gambar 5.52 Implementasi Antarmuka Halaman Login *Increment* 2 Status Gagal

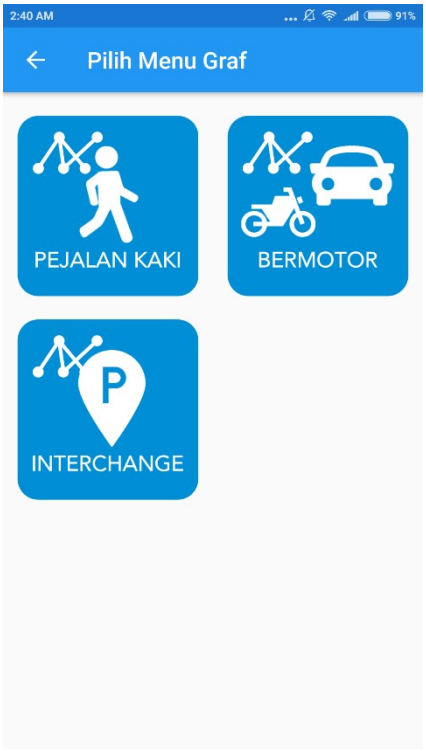
Pada implementasi antarmuka halaman login, pengguna dapat memasukkan email dan password melalui form input berupa Edit Text yang ada. Setelah itu pengguna melakukan *tap* terhadap tombol *login* untuk memulai proses *login*. Proses *login* berlangsung ditandai dengan munculnya Progress Bar. Setelah itu pengguna akan dialihkan ke halaman utama jika proses *login* berhasil, dan akan mendapatkan notifikasi jika proses *login* gagal.

5.2.8.2 Implementasi Antarmuka Halaman Utama

Gambar 5.53 Implementasi Antarmuka Halaman Utama *Increment* 2

Pada implementasi antarmuka halaman utama *increment* 2, terdapat 2 pilihan opsi yang dapat dipilih oleh pengguna, yaitu pilihan untuk mengedit graf dan pilihan untuk melakukan navigasi menggunakan graf yang tersimpan pada database. Selain itu, pengguna juga bisa mengakses halaman Setting dan Logout melalui Menu Button pada pojok kiri atas.

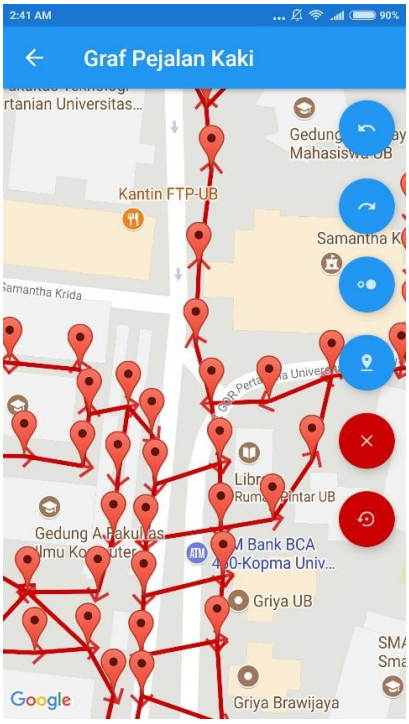
5.2.8.3 Implementasi Antarmuka Halaman Pilih Tipe Graf Untuk Edit



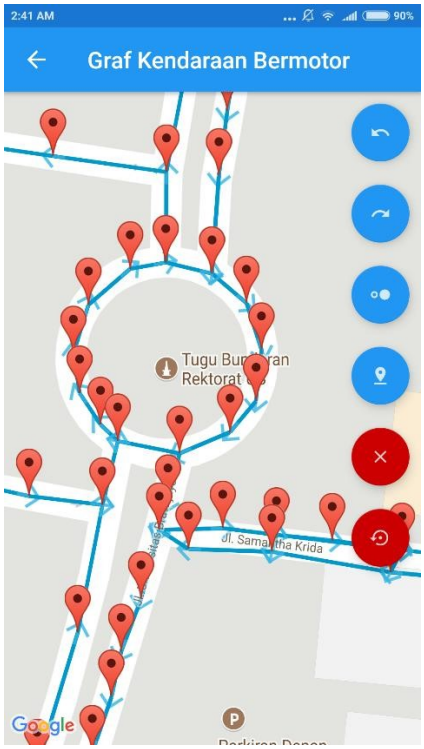
Gambar 5.54 Implementasi Antarmuka Halaman Pilih Tipe Graf Untuk Edit *Increment* 2

Pada implementasi antarmuka halaman tipe graf untuk edit *increment* 2, pengguna dapat memilih data yang akan di edit. Terdapat 3 pilihan data, yaitu data graf pejalan kaki, data graf kendaraan bermotor, dan data lokasi *interchange*. Pengguna dapat memilih opsi dengan melakukan *tap* pada gambar yang ada. Pengguna dapat kembali ke halaman sebelumnya dengan melakukan *tap* pada tombol Back Button di Action Bar atau menggunakan tombol Back fisik.

5.2.8.4 Implementasi Antarmuka Halaman Edit Graf



Gambar 5.55 Implementasi Antarmuka Halaman Edit Graf Pejalan Kaki *Increment* 2



Gambar 5.56 Implementasi Antarmuka Halaman Edit Graf Kendaraan Bermotor *Increment* 2

Pada implementasi antarmuka halaman edit graf *increment* 2, terdapat 2 kemungkinan yang terjadi, yaitu pengguna memilih graf pejalan kaki atau graf kendaraan bermotor. Tampilan halaman di dominasi oleh tampilan peta Google Maps, yang didalam nya terdapat kumpulan titik yang ditampilkan sebagai Marker dan kumpulan jalur yang ditampilkan sebagai Polyline. Titik dan jalur didapat berdasarkan pilihan graf yang dipilih oleh pengguna.

Pada graf pejalan kaki, jalur graf akan berwarna merah dan pada graf kendaraan bermotor, jalur graf akan berwarna biru, hal ini dilakukan untuk membedakan tampilan antara graf pejalan kaki dan graf kendaraan bermotor. Di sisi kanan layar terdapat panel yang berisi tombol-tombol pendukung yaitu tombol *undo*, tombol *redo*, tombol *join*, tombol *add*, dan tombol hapus, dan tombol *reset*. Tombol-tombol tersebut berfungsi untuk mendukung pengguna dalam melakukan perubahan pada graf yang diplih. Fungsi tombol-tombol tersebut adalah :

Tombol *undo* : Mengembalikan proses yang telah dilakukan sebanyak 1 langkah

Tombol *redo* : Mengembalikan proses yang dilakukan undo sebanyak 1 langkah

Tombol *join* : Menyatukan 2 titik

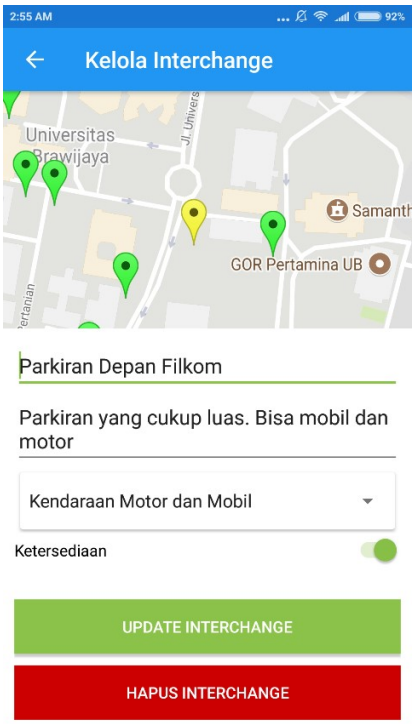
Tombol *add* : Menambah titik diluar graf yang telah dibuat

Tombol hapus : Menghapus suatu titik

Tombol *reset* : Menghapus seluruh titik dan jalur yang terdapat dalam suatu graf

5.2.8.5 Implementasi Antarmuka Halaman Edit *Interchange*

Gambar 5.57 Implementasi Antarmuka Halaman Edit Interchange *Increment* 2 Status Menambah *Interchange*



Gambar 5.58 Implementasi Antarmuka Halaman Edit Interchange *Increment* 2 Status Mengedit *Interchange*

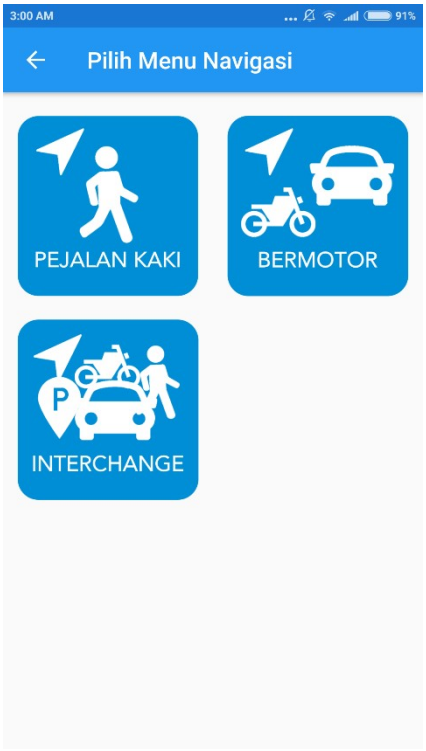
Pada implementasi antarmuka halaman edit interchange *increment* 2, pengguna dapat membuat, mengedit, serta menghapus *interchange* di halaman ini. Halaman ini di dominasi oleh tampilan peta Google Maps yang digunakan oleh pengguna untuk berinteraksi.

Untuk memasukkan lokasi *interchange* baru, pengguna melakukan *tap* pada area peta, hal ini akan memunculkan Bottom Sheet yang berisi form untuk *interchange* baru. Setelah *form* diisi dengan benar, pengguna melakukan *tap* pada tombol Submit untuk menyimpan data *interchange* baru ke *database*.

Untuk mengedit data lokasi *interchange*, pengguna dapat menggeser Marker *interchange* yang akan diubah lokasinya ke lokasi yang baru. Sedangkan untuk mengedit data penjelasan *interchange*, pengguna melakukan *tap* pada Marker yang akan diubah, hal ini akan memunculkan Bottom Sheet yang telah terisi data *interchange* terpilih yang dapat diganti dengan data baru oleh pengguna. Setelah *form* diisi dengan benar, pengguna melakuakan *tap* pada tombol Update untuk menyimpan data *interchange* terbaru ke *database*.

Untuk menghapus interchange, pengguna melakukan *tap* pada Marker yang akan diubah, hal ini akan memunculkan Bottom Sheet yang telah terisi data *interchange* terpilih. Setelah itu pengguna melakukan *tap* pada tombol Delete untuk menghapus *interchange* dari *database*.

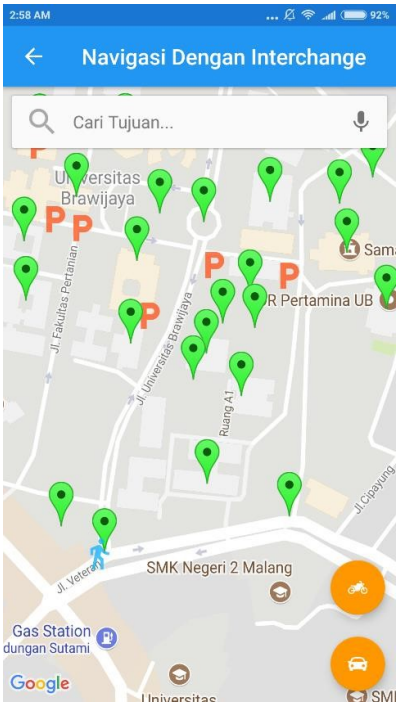
5.2.8.6 Implementasi Antarmuka Halaman Pilih Tipe Graf Untuk Navigasi



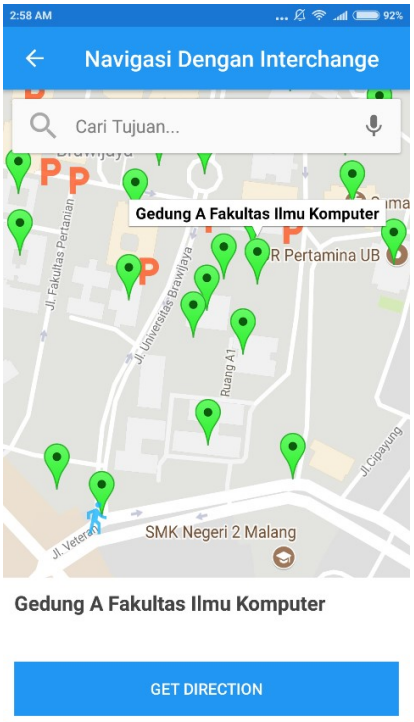
Gambar 5.59 Implementasi Antarmuka Halaman Pilih Tipe Graf Untuk Navigasi *Increment* 2

Pada implementasi antarmuka halaman tipe graf untuk navigasi *increment* 2, pengguna dapat memilih jenis navigasi yang akan dilakukan. Terdapat 3 pilihan navigasi, yaitu data navigasi dengan graf pejalan kaki, navigasi dengan graf kendaraan bermotor, dan navigasi dengan gabungan graf pejalan kaki dan graf kendaraan bermotor menggunakan *interchange*. Pengguna dapat memilih opsi dengan melakukan *tap* pada gambar yang ada. Pengguna dapat kembali ke halaman sebelumnya dengan melakukan *tap* pada tombol Back Button di Action Bar atau menggunakan tombol Back fisik.

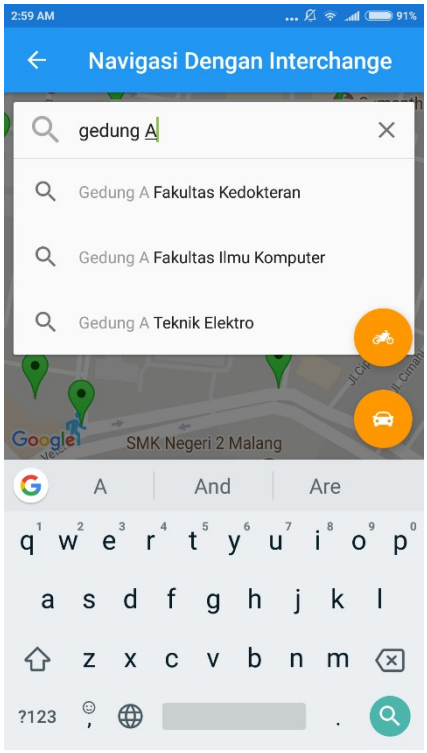
5.2.8.7 Implementasi Antarmuka Halaman Navigasi



Gambar 5.60 Implementasi Antarmuka Halaman Navigasi *Increment* 2



Gambar 5.61 Implementasi Antarmuka Halaman Navigasi *Increment* 2 Status Memilih Lokasi Tujuan



Gambar 5.62 Implementasi Antarmuka Halalaman Navigasi *Increment* 2 Status Mencari Lokasi Tujuan

Gambar 5.63 Implementasi Antarmuka Halalaman Navigasi *Increment* 2 Status Menampilkan Rute Terdekat

Pada implementasi antarmuka halaman navigasi *increment* 2, pengguna dapat melakukan proses perhitungan rute terdekat ke lokasi di wilayah Universitas Brawijaya berdasarkan jenis navigasi yang dipilih sebelumnya. Halaman di dominasi dengan tampilan peta Google Maps yang digunakan oleh pengguna untuk berinteraksi.

Pada tampilan Google Maps, terdapat Marker yang menunjukkan posisi awal pengguna yang ditampilkan dengan *icon* manusia berwarna biru. Marker posisi awal pengguna dapat dipindahkan dengan melakukan *tap* pada area pada peta Google Maps. Selain itu, tampilan Google Maps juga berisi Marker yang merupakan lokasi-lokasi tujuan yang dapat dipilih di area Universitas Brawijaya.

Melakukan *tap* pada Marker lokasi tujuan akan memunnculkan Bottom Sheet yang menampilkan informasi tujuan dan tombol untuk memunculkan rute terdekat yang dapat di *tap* oleh pengguna untuk memunculkan jalur rute terdekat berdasarkan jenis navigasi yang dipilih oleh pengguna sebelumnya.

Selain melalui tombol pada Bottom Sheet, pengguna juga dapat memunculkan jalur rute terdekat dengan memanfaatkan Search View yang terdapat di atas tampilan peta. Pengguna dapat memasukkan nama lokasi yang dituju, dan melakukan *tap* pada hasil pencarian untuk memunculkan rute terdekat ke lokasi tersebut berdasarkan jenis navigasi yang dipilih oleh pengguna sebelumnya.

Pada jenis navigasi gabungan graf pejalan kaki dan graf kendaraan bermotor menggunakan *interchange*, terdapat 2 tombol baru di bawah kanan yang akan melakukan *filter* untuk jenis *interchange* yang dapat digunakan dan ditampilkan. Pilihan *interchange* yang dapat digunakan oleh pengguna adalah *interchange* kendaraan motor, *interchange* kendaraan mobil, dan *interchange* kendaraan motor dan mobil.